Entwicklung eines Konzeptes für die Optimierung von Produktentwicklungsprojekten auf Basis eines integrierten Produktdaten- und Projektmanagements

Der Fakultät für Ingenieurwissenschaften, Abteilung Maschinenbau der
Universität Duisburg-Essen
zur Erlangung des akademischen Grades

DOKTOR-INGENIEUR

genehmigte Dissertation

von

Dominik Raab
aus
Oberhausen

Referent:	Priv.- Doz. Dr.-Ing. Frank Lobeck
Korreferent:	Prof. Dr.-Ing. Diethard Bergers
Tag der mündlichen Prüfung:	04.12.2008

Vorwort

Diese Arbeit entstand im Zeitraum von Januar 2006 bis Juni 2008 während meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Fakultät für Ingenieurwissenschaften der Universität Duisburg-Essen.

Herrn Prof. Dr.-Ing. H. J. Stracke und Herrn Priv.-Doz. Dr.-Ing. F. Lobeck danke ich für die Anregung und Förderung dieser Arbeit und den mir stets gewährten Freiraum bei der Durchführung. Zusätzlich danke ich Herrn Priv.-Doz. Dr.-Ing. F. Lobeck für die konstruktiven Diskussionen und die Übernahme des Erstgutachtens.

Herrn Prof. Dr.-Ing. D. Bergers danke ich für das meiner Arbeit entgegengebrachte Interesse und die Übernahme des Korreferats. Ebenso danke ich Herrn Prof. Dr.-Ing. Dipl.-Math. Peter Köhler für die Übernahme des Prüfungsvorsitzes.

Danken möchte ich auch meinem Kollegen Dr.-Ing. T. Strassmann für seine Bereitschaft, meine Konzepte mit ihm diskutieren zu können und die Anregungen, die aus unserer gemeinsamen Arbeit hervorgegangen sind.

Einen speziellen Dank möchte ich an dieser Stelle auch meiner Freundin Lydia für die Unterstützung und das Verständnis aussprechen, welche sie mir während der Erstellung dieser Arbeit entgegengebracht hat.

Weiterhin gilt ein ganz besonderer Dank meinen Eltern, die mir meinen beruflichen Werdegang ermöglichten und mich in allen beruflichen und privaten Belangen stets begleitet haben.

Abschließend möchte ich mich an dieser Stelle auch bei allen namentlich nicht genannten Kollegen und Freunden bedanken, die mich im Laufe meiner Promotion unterstützt haben.

Essen, im Juni 2008

Dominik Raab

Inhaltsverzeichnis

1 Einleitung und Motivation

Durch die zunehmende Globalisierung und eine verschärfte Wettbewerbssituation sind produzierende Unternehmen einem ständig steigenden Konkurrenzdruck ausgesetzt. Daher ist es für den Erfolg eines produzierenden Unternehmens von essentieller Bedeutung, fortlaufend neue und innovative Produkte auf dem Markt zu positionieren.
Ferner ist bei der Komplexität der Produkte ein stetiger Anstieg zu verzeichnen, während sich die Zeiträume in denen ein Produkt gewinnbringend vermarktet werden kann in den letzten 20 Jahren in etwa halbiert haben [1].
Aus diesen Gründen bestehen für die überwiegende Mehrheit der produzierenden Unternehmen wesentliche strategische Ziele in der Optimierung der Produktentwicklung und der Minimierung des Zeitraums zwischen der ersten Produktidee und der Markteinführung (Time-To-Market).

Die Grundvoraussetzung für die Entwicklung und Herstellung von innovativen Produkten in möglichst kurzen Zeiträumen besteht in der Anwendung des Simultaneous Engineerings. Dieser Begriff umschreibt die gleichzeitige Bearbeitung von unterschiedlichen Arbeitsschritten durch verschiedene Mitarbeiter oder Abteilungen. Im Gegensatz zu der konventionellen, sequentiellen Arbeitsweise kann durch diese Technik eine wesentlich kürzere Time-To-Market erzielt werden (vgl. Abbildung 1-1).

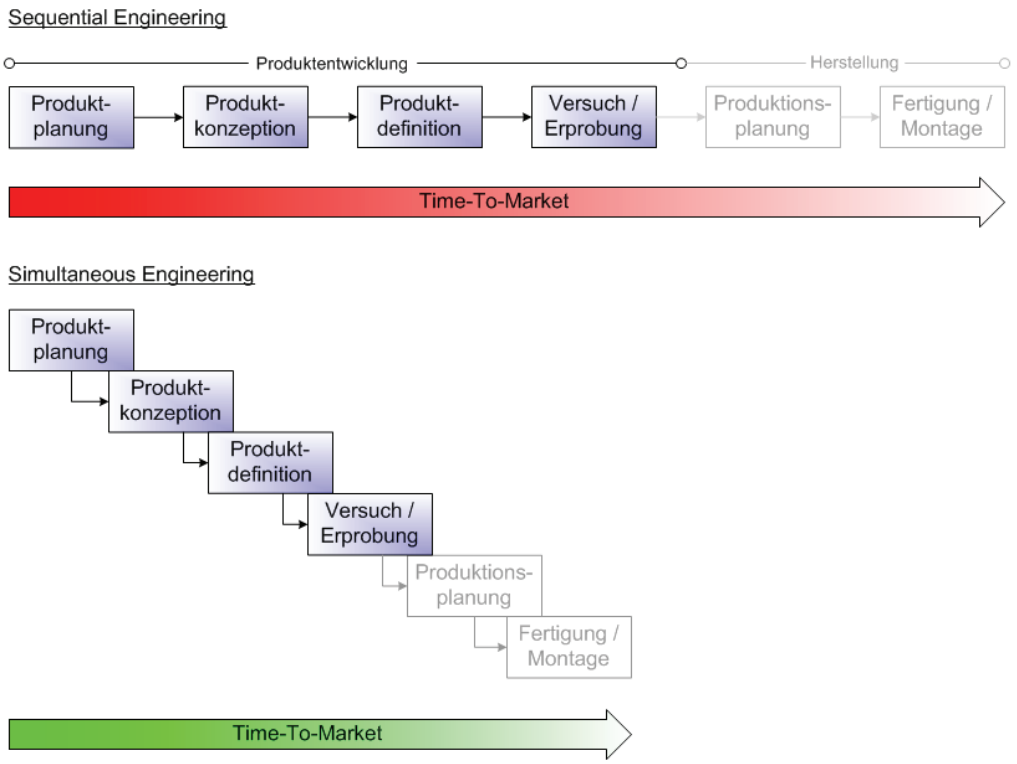


Abbildung 1-1: Sequential vs. Simultaneous Engineering [2]

Die Realisierung des Simultaneous Engineerings bedarf des Einsatzes eines modernen EDV-Systems, welches den für die Parallelisierung erforderlichen Informationsfluss steuert und eine einheitliche Datenverwaltung verwirklicht. Diese Funktionen werden von einem Produktdatenmanagement-System zur Verfügung gestellt.
Das PDM-System verwaltet die Gesamtheit aller digitalen Informationen, die zur Beschreibung eines Produktes erforderlich sind und dient als Integrationsplattform für die heterogene Landschaft von produktdatenerzeugenden EDV-Systemen. Sämtliche an der Produktentstehung beteiligte Abteilungen können somit auf ein zentrales, konsistentes Produktmodell zugreifen. Dadurch werden Fehlergefahr und Überarbeitungsaufwand um *bis zu 50%* gesenkt [3].
Ferner stellt jedes PDM-System eine Prozessmanagement-Komponente zur Verfügung, welche die Abläufe bei der Datenerstellung abbildet und steuert. Somit wird der Informationsfluss bei der Produktentstehung koordiniert und eine transparente Organisation der Arbeitsabläufe bei der Datenerstellung realisiert.
Aus der Durchführung des Simultaneous Engineerings mit Unterstützung eines PDM-Systems resultiert eine Verkürzung der Time-To-Market von bis zu 25% [4].

Innerhalb des Produktentstehungsprozesses bedarf der Produktentwicklung eine besondere Aufmerksamkeit. Untersuchungen zufolge werden etwa 75% der Herstellungskosten während der Produktentwicklung bestimmt [5]. Weiterhin ist es erwiesen, dass nach Abschluss der Entwicklungsphase Fehler nur unter hohem Zeit- und Kostenaufwand behoben werden können.
Eine Verkürzung der Produktentwicklung, ohne gleichzeitige qualitative Verschlechterung, stellt eine große strategische Herausforderung für heutige Unternehmen dar. Um dem Anspruch nach immer kürzeren Entwicklungszeiten bei gleichzeitig kostengünstigen und innovativen Produkten gerecht zu werden, müssen sowohl die Abhängigkeiten zwischen den einzelnen Entwicklungsschritten als auch die komplexen Wirkungszusammenhänge zwischen der benötigten Zeit, dem investierten Geld und der erzielten Qualität berücksichtigt werden.
Projektmanagement-Systeme fördern eine *organisierte Führung, Koordination, Steuerung und Kontrolle* von Entwicklungsvorhaben in Form von Projekten [6]. Der Einsatz eines PM-Systems ermöglicht es, die Zusammenhänge zwischen den einzelnen Teilvorgängen eines Projektes zu analysieren und unter Berücksichtigung der Wechselwirkung zwischen den drei Haupteinflussfaktoren Zeit, Qualität und Geld einen Projektplan zu modellieren, der allen Zielvorgaben gerecht wird.

Im Zusammenspiel mit einem integrierten Werkzeug zum Projektcontrolling[1] lässt sich durch ein Projektmanagement-System *eine Reduzierung der Projektdauer und des Aufwandes um bis zu 20%"* erzielen [7]. Der Einsatz von PM-Systemen bildet infolgedessen eine fundamentale Voraussetzung für eine wettbewerbsfähige Produktentwicklung.

Zusammenfassend werden in Fertigungsunternehmen mittels EDV-Systemen zwei wesentliche Ansätze zur Optimierung der Produktentwicklung und der Verkürzung der Time-To-Market betrieben:

- Produktdatenmanagement (PDM-System)
- Projektmanagement (PM-System)

Das Potenzial von PDM- und PM-Systemen wird bereits mit einem relativ hohen Reifegrad ausgeschöpft. Für beide Technologien wird auf dem Markt eine breite Palette an Anwendungen mit weit gefächertem Funktionsumfang angeboten.
Das zentrale Hindernis für die Realisierung eines effizienten Produktentwicklungsprojektes besteht in der mangelnden Synchronisation zwischen dem Produktdatenmanagement und dem Projektmanagement. Es existiert keine durchgängige EDV-Lösung. Stattdessen werden die Planung und Steuerung des Projektes mit Hilfe eines PM-Systems ausgeführt, während die Durchführung der Produktentwicklung unter Verwendung eines PDM-Systems erfolgt. Der einzige Datenaustausch, der in der Regel zwischen diesen beiden Systemen praktiziert wird, besteht in einem manuellen Datenabgleich. Dieses Verfahren ist sowohl zeitaufwendig als auch fehleranfällig, da von keinem der beiden Systeme geprüft werden kann, ob notwendige Änderungen berücksichtigt wurden.

Ein weiteres nicht zu unterschätzendes Problem für die Durchführung einer effizienten Produktentwicklung besteht darin, dass keine zentrale Benutzeroberfläche zur Verfügung steht, welche die kombinierten Informationen aus dem PDM- und dem PM-System übersichtlich aufbereitet.
Es gibt im PDM-System zwar Funktionen um den Workflow einzelner Prozesse abzubilden, eine Gesamtübersicht über das Projekt fehlt aber. Das PM-System bildet zwar das Gesamtprojekt ab, bietet aber keinen Zugriff auf die Projektdaten und ist dadurch von allen wesentlichen Informationen über das Entwicklungsprojekt abgeschnitten. Die Projektleitung besitzt daher keine Möglichkeit schnelle Entscheidungen auf Grundlage sämtlicher relevanten

[1] Controlling: <engl. Steuerung> Steuerungs- und Koordinationskonzept zur ergebnisorientierten Planung und zielsetzungsgerechter Entscheidungsfindung, umfasst die strukturierte Beschaffung, Aufbereitung und Analyse von Daten [6]

Informationen zu treffen oder die Auswirkungen einer Entscheidung detailliert auszuwerten. Ferner resultiert aus dieser Sachlage ein Informationsdefizit für Auftraggeber und Projektmitglieder. Ihnen erschließen sich jeweils nur Teilaspekte des Projektes.
Diese Faktoren führen in ihrer Gesamtheit dazu, dass das Optimierungspotenzial, welches lokal durch den Einsatz von effizienten PDM- und PM-Systemen erschlossen wird, global zu einem beträchtlichen Teil wieder vernichtet wird.

In Anbetracht der dargelegten Gründe soll im Rahmen dieser Arbeit ein Konzept für ein vollständig integriertes Produktdaten- und Projektmanagement entwickelt werden.

Die Erarbeitung des Konzeptes wird in mehreren Teilschritten vollzogen. Zunächst werden die verfügbaren PDM- und PM-Systeme einer kritischen Betrachtung unterzogen und die vorhandenen Defizite herausgestellt (Kapitel 2). Nachfolgend werden aus den gewonnenen Erkenntnissen die Anforderungen an ein neues Konzept abgeleitet und die technologischen Randbedingungen für die Umsetzung dargelegt (Kapitel 3 und 4). Im Anschluss daran wird die Konkretisierung des Konzeptes vollzogen (Kapitel 5), bevor eine Verifizierung anhand einer beispielhaften Realisierung erfolgt (Kapitel 6).

2 Stand der Technik

Als Grundlage für die Entwicklung eines integrierten Produktdaten- und Projektmanagements werden in diesem Kapitel Produktdatenmanagement und Projektmanagement einer eingehenden Betrachtung unterzogen.
Einleitend werden jeweils die wesentlichen Prinzipien und Methoden der beiden Managementstrategien dargelegt, bevor eine Betrachtung des Stands der Technik der dazugehörigen EDV-Systeme durchgeführt wird. Dabei werden sowohl Einzelsysteme zum Produktdaten- und Projektmanagement thematisiert als auch kombinierte Systeme behandelt.
Abschließend erfolgt schließlich eine Analyse der Defizite der heutigen Situation, die als Grundlage für die Erstellung eines Anforderungskataloges dient.

2.1 Produktdatenmanagement

2.1.1 Grundlagen des Produktdatenmanagements

Für das Produktdatenmanagement gibt es keine DIN-Norm, die eine einheitliche Begriffsdefinition festlegt. In der Fachliteratur werden daher oftmals dieselben Begriffe für unterschiedliche Inhalte verwendet. Weiterhin existiert für PDM-Systeme eine Vielzahl von gängigen Synonymen. Engineering Database (EDB), Engineering Data Management (EDM), Technisches Informationssystem (TIS), Product Information Management (PIM), Technical Data Management (TDM) sowie Product Lifecycle Management (PLM), Virtual Product Development (VPDM) und Product Development Management (PDM2) sind als Bezeichnungen für Systeme mit PDM-Funktionalität gebräuchlich. Diese Begriffe bezeichnen alle die gleiche Art von Systemen, die sich in ihrer Funktionalität nur unwesentlich voneinander differenzieren. Die unterschiedlichen Bezeichnungen sollen lediglich verdeutlichen, dass die neuen Generationen von PDM-Systemen in zunehmendem Maße zur Unterstützung des gesamten Lebenszyklus eines Produktes geeignet sind. Dies umfasst das Informations- und Datenmanagement von der ersten Produktidee bis zur Entsorgung. [8]
Im Rahmen dieser Arbeit wird keine Unterscheidung zwischen den oben aufgeführten Ausprägungen des Produktdatenmanagements vorgenommen. Es wird im Folgenden der Begriff PDM entsprechend der nachstehenden Definition verwendet.

Produkt Daten Management oder PDM bezeichnet die ganzheitliche Verwaltung aller Daten, die bei der Entwicklung neuer Produkte oder der Aktualisierung bisheriger Produkte anfallen, bearbeitet und weitergeleitet werden müssen, verbunden mit der Fähigkeit, den Prozess der Bearbeitung und Weiterleitung zu steuern und zu kontrollieren.

Auf dieser Definition aufbauende Produktdatenmanagement-Systeme oder PDMS sind Rechnersysteme zur Verwaltung technischer Daten und Dokumente von Produkten. Weiterhin fungieren PDM-Systeme als *Integrationsplattformen (...)*, die während der Produktentwicklung benötigte *Applikationssysteme (z. B. CAx-Systeme, Office-Programme, NC-Tools) über Schnittstellen zu einem Gesamtsystem verbinden*. [9] [10]

Entsprechend dieser Definition umfasst das Produktdatenmanagement drei Kernfunktionen:
- Datenmanagement
- Prozessmanagement
- Integration

Datenmanagement

Eine zentrale Aufgabe eines PDM-Systems ist die Verwaltung der ständig wachsenden Menge an digitalen Dokumenten eines Unternehmens. Ziel ist es, allen Mitarbeitern den Zugriff auf alle im Unternehmen existierende Dokumente in Abhängigkeit ihrer Zugriffsberechtigungen zu ermöglichen.
Das Datenmanagement setzt sich aus folgenden Teilbereichen zusammen:

• Dokumentenmanagement

In diesen Funktionsbereich fallen die konsistente, unternehmensweite Bereitstellung aller Dokumente sowie die Kopplung mit dem jeweiligen Erzeugersystem.
Dabei wird das schnelle Auffinden von Dokumenten durch übersichtliche Suchmasken unterstützt. Die Anwender müssen daher nicht mehr wissen wo gesuchte Dokumente physikalisch gespeichert sind oder mit welcher Anwendung sie erzeugt wurden.

• Klassifizierung

Die sachbezogene Ablage von Objekten wird als Klassifizierung bezeichnet. Die Klassifizierung wird innerhalb eines PDM-Systems mit Metadaten realisiert. Dabei handelt es sich um beschreibende Daten, die mit einzelnen Dokumenten, Teilen und Produkten zu einem Stammdatensatz verschmolzen werden. Metadaten ermöglichen die Bildung logischer Informationseinheiten und sind Grundvoraussetzung für die Handhabung von Teilefamilien. Weiterhin beruhen die Such- und Zugriffsmechanismen des PDM-Systems auf der Klassifizierung mit Metadaten.

• Versionsmanagement

Neben der aktuellen Version aller Dokumente verwaltet ein PDM-System auch deren gesamte Änderungshistorie. Das Versionsmanagement legt dabei die Beziehungen zwischen den aktuellen Daten und den Vorgängerversionen fest.

• Produktstruktur- und Konfigurationsmanagement

Das Produktstruktur- und das Konfigurationsmanagement umfassen die zentralen Funktionen, die zum Aufbau eines virtuellen Produktmodells erforderlich sind.

Die Produktstruktur definiert mittels Verknüpfungen aus welchen produktbeschreibenden Daten und Dokumenten sich ein virtuelles Produkt zusammensetzt. Währendessen werden in Form von Konfigurationen die unterschiedlichen Varianten der virtuellen Produkte verwaltet.

Wichtige Funktionen des Produktstruktur- und Konfigurationsmanagements sind der Verwendungsnachweis und der Referenzierungsnachweis. Diese Nachweise geben Aufschluss darüber von welchen Produktstrukturen ein bestimmtes Bauteil verwendet wird bzw. von welchen Produktstrukturen ein bestimmtes Dokument referenziert wird. Auf diese Weise kann bei der Änderung von Bauteilen und Dokumenten schnell geprüft werden, welche Produkte bzw. Konfigurationen von der Änderung betroffen sind.

Prozessmanagement

Die zweite Kernfunktion des Produktdatenmanagements ist die Organisation der Arbeitsabläufe bei der Datenerstellung. Dies umfasst die Abbildung aller technischen und organisatorischen Abläufe, die im Laufe eines Produktlebens anfallen und die aktive Steuerung des dazugehörigen Datenflusses. Das Prozessmanagement konzentriert sich dabei weniger auf das reine Delegieren von Aufgaben, sondern es steht die Wechselwirkung zwischen Aufgaben und Daten im Vordergrund. [11]

Das Prozessmanagement umfasst folgende Teilaufgaben:

• Workflowmanagement

Ein Geschäftsprozess ist eine logische Abfolge von einzelnen Arbeitsaktivitäten. Die rechnergestützte Abwicklung von Geschäftsprozessen wird als Workflow[2] bezeichnet. Das Workflowmanagement koordiniert die durchzuführenden Arbeiten nach einem vordefinierten Ablaufschema und reguliert den Datenfluss, indem es vorliegende Arbeitsergebnisse automatisch an den nachfolgenden Prozessschritt weiterleitet.

• Verteil- und Informationssystem

Die herkömmliche Verteilung von Unterlagen wird durch den Einsatz eines PDM-Systems einem grundlegenden Wandel unterzogen. Die ausgearbeiteten Daten und Dokumente werden nach Freigabe von dem PDM-System zum richtigen Zeitpunkt an der benötigten Stelle zur Verfügung gestellt. Alternativ werden die betroffenen Stellen per Email-Verteiler darüber informiert, dass neue oder geänderte Daten zur Verfügung stehen.

---

[2] Workflow: <engl.> Arbeitsablauf [12]

• Änderungs- und Freigabemanagement

Modifikationen an bereits freigegebenen Objekten werden als Änderungen verstanden. Organisatorisch ist vor jeder Änderung zu prüfen, inwieweit sie sich auf andere Strukturstufen des Produktes und auf andere Produkte auswirkt. Dies lässt sich in der Regel nicht automatisiert durchführen, so dass menschliche Entscheidungsträger erforderlich sind, welche die entsprechenden Änderungen freigeben.

Das workflowgesteuerte Änderungs- und Freigabemanagement gewährleistet, dass nur fachlich einwandfreie und für die Produktstruktur unbedenkliche Dokumente in Umlauf gelangen.

Weiterhin wird im Zuge des Produktdatenmanagements die gesamte Änderungshistorie eines Produktes lückenlos dokumentiert, so dass die strengen Standards des Qualitätsmanagements nach den Normen der DIN EN ISO 9000-Reihe erfüllt werden.

• Benutzermanagement und Zugriffsverwaltung

Um Datenschutz und Datensicherheit zu gewährleisten muss sichergestellt sein, dass ausschließlich berechtigte Personen gespeicherte Daten lesen und modifizieren können. Dies erfordert ein Benutzermanagement und eine Zugriffsverwaltung. Das Benutzermanagement stellt Funktionen bereit, die es erlauben eine Organisationsstruktur, bestehend aus Benutzer, Benutzergruppen und Rollen, im System abzubilden. Darauf aufbauend ermöglicht die Zugriffsverwaltung die Vergabe von spezifischen Zugriffsrechten. Die Regelung des Zugriffs ist dabei eng mit dem Status eines Dokumentes verbunden.

Integration

Eine weitere zentrale Funktion eines PDM-Systems besteht darin als Integrationsplattform für die Vielzahl von im Unternehmen vorhandenen datenerzeugenden Systemen zu dienen. Diese Verknüpfung aller an der Produktentwicklung beteiligten EDV-Systeme ist Grundlage für die Erstellung eines virtuellen Produktmodells, das alle produkt- und produktionsbeschreibenden Daten enthält und das alle beteiligten Fachbereiche nutzen, ergänzen und auch modifizieren können.

2.1.2 Produktdatenmanagement-Systeme

Aufgrund des breit gefächerten Angebots an PDM-Systemen kann aus Platzgründen an dieser Stelle nicht auf spezifische Systeme eingegangen werden. Stattdessen wird nachfolgend der grundlegende Aufbau von PDM-Systemen anhand ihrer typischen Systemarchitektur erläutert.
Abbildung 2-1 zeigt die Architektur, die den meisten PDM-Systemen zugrunde liegt. Abweichungen von dieser Architektur sind entweder durch spezielle Anwendungsschwerpunkte oder die historische Entwicklung des betreffenden PDM-Systems bedingt.

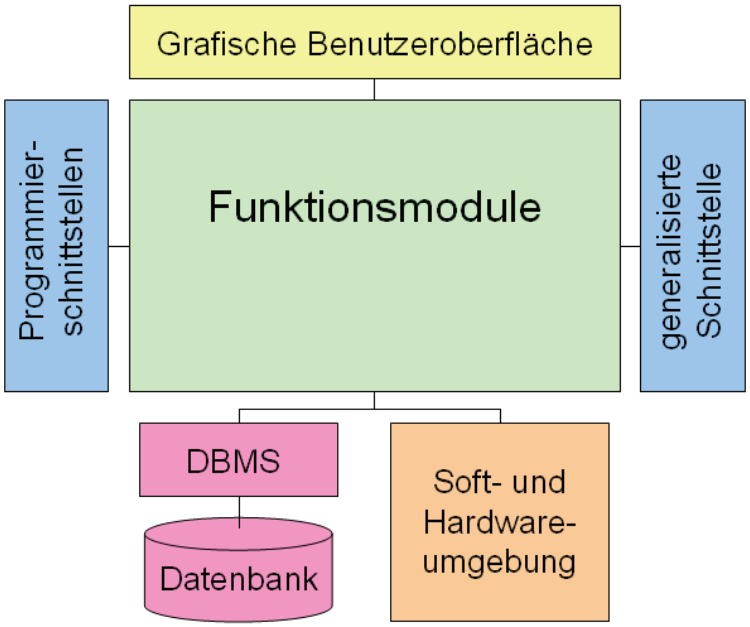


Abbildung 2-1: PDM Systemarchitektur [11] [13] [14]

Im Einzelnen stehen bei einem typischen PDM-System folgende Komponenten in Wechselwirkung:

Grafische Benutzeroberfläche

Die Benutzeroberfläche ist die Schnittstelle zwischen Anwender und System und muss daher die gesamte Funktionalität des PDM-Systems abbilden.

Funktionsmodule

Das breite Funktionsspektrum eines PDM-System lässt sich am effizientesten mittels eines modular aufgebauten Systems implementieren. Der in Kapitel 2.1.1 dargelegte Funktionsumfang eines PDM-Systems ist daher in unterschiedliche Funktionsbereiche untergliedert, die jeweils in einem eigenen Modul implementiert sind. Nach dem Pyramidenprinzip bauen Module mit steigender Komplexität auf mehrere Untermodule auf.

Programmierschnittstelle

Programmierschnittstellen ermöglichen die Erweiterung der bestehenden Funktionalität des PDM-Systems durch die Integration weiterer Funktionsmodule.

Weiterhin dient die Programmierschnittstelle der Anpassung des PDM-Systems an kundenspezifische Anforderungen, dem sogenannten Customizing.

Generalisierte Schnittstelle

Über die generalisierte Schnittstelle erfolgt ein standardisierter Datenaustausch zwischen dem PDM-System und den datenerzeugenden Systemen. Durch die Generalisierung der Schnittstelle ist gewährleistet, dass die Anbindung neuer EDV-Systeme vergleichsweise wenig Programmieraufwand verursacht.

Soft- und Hardwareumgebung

Die Soft- und Hardwareumgebung besteht aus der Hardware, einem Betriebssystem und einem Netzwerkprotokoll zur Kommunikation mit Fremdrechnern.

Datenbank und DBMS

Das PDM-System muss in der Lage sein, die gesamten technischen Daten des Unternehmens, unter ständiger Verfügbarkeit und einem hohen Maß an Sicherheit, zu verwalten. Dies wird durch eine Datenbank realisiert, deren Nutzung mit einem leistungsfähigen Datenbank-Managementsystem (DBMS) gesteuert wird. Der Datenbestand des PDM-Systems kann dabei sowohl in einer zentralen Datenbank erfolgen oder auf mehrere Datenbanken verteilt werden.

PDM-Systeme werden hauptsächlich nach dem Client-Server-Prinzip betrieben. Durch den modularen Systemaufbau können, den Anforderungen des Unternehmens entsprechend, viele Funktionen des Systems auf einem zentralen Server aufgesetzt werden. Wird eine Minimierung der Kosten von Lizenzen und Client-Hardware angestrebt, so lassen sich sämtliche Anwendungsmodule auf den Server auslagern. Somit ist auf den Clients nur noch die Benutzeroberfläche erforderlich. Stehen hingegen Performanceanforderungen im Vordergrund kann die komplette PDM-Funktionalität auf den Clients bereitgestellt werden. In diesem Fall dient der Server primär der Verwaltung der Datenbank.

2.2 Projektmanagement

2.2.1 Grundlagen des Projektmanagements

Da der Begriff 'Projekt' im allgemeinen Sprachgebrauch eine Vielzahl synonymer Verwendungen findet, ist vor der Darlegung der Grundlagen des Projektmanagements zunächst eine eindeutige Definition des Projektbegriffes erforderlich.

Die DIN 69901 bezeichnet ein Projekt als ein *Vorhaben, das im wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist*.
Diese Definition grenzt Projekte aufgrund ihrer Einmaligkeit klar von Routineaufgaben ab. Eine weiterführende Begriffsdefinition von Rinza versucht die Einmaligkeit der Bedingungen eines Projektes aufzuschlüsseln, die anhand der DIN-Norm nur schwer greifbar ist: *Ein Projekt wird definiert als ein Vorhaben, dessen Ablauf zumindest weitgehend einmalig ist, dessen Struktur eine bestimmte Komplexität aufweist* und *dessen festgelegte Zielsetzung in vorgegebener Zeit mit den gegebenen Mitteln zu erreichen ist* [15].
In einer Definition von Seibert wird zudem die Abgeschlossenheit als zentrale Eigenschaft berücksichtigt: *Projekte sind Vorhaben mit definiertem Anfang und Abschluss* [16].

Auf Grundlage dieser Definitionen wird im Rahmen dieser Arbeit ein Projekt als ein hinreichend komplexes Vorhaben verstanden, welches sich dadurch auszeichnet, dass es weitgehend einmalig ist und eine zeitliche Begrenzung aufweist.
Da das Projektziel mittels endlicher Ressourcen zu einem festgelegten Endtermin in einer bestimmten Qualität realisiert werden muss, sind Qualität, Termine und Kosten als zentrale Randbedingungen eines jeden Projektes anzusehen. Die folgende Abbildung 2-2 verdeutlicht, dass diese Größen in einem wechselseitigen Abhängigkeitsverhältnis stehen. Mit einer Reduzierung der Kosten geht beispielsweise zwangsläufig eine Verspätung der Termine und oder eine Verschlechterung der Qualität einher.

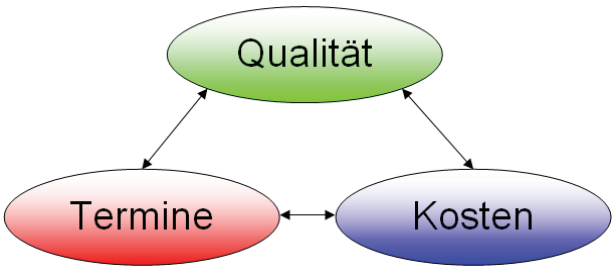


Abbildung 2-2: Magisches Dreieck des Projektmanagements [16]

Neben diesen drei zentralen Randbedingungen wird jedes Projekt durch zahlreiche Nebenbedingungen geprägt. Dazu gehören gesetzliche Bestimmungen, wirtschaftliche Einflussgrößen, technische Restriktionen und vieles mehr.
Aufgrund der komplexen Wirkzusammenhänge zwischen den einzelnen Haupt- und Nebenbedingungen sind zur ganzheitlichen Planung, Überwachung und Steuerung von Projekten die Methoden des Projektmanagements unerlässlich.
Der Begriff Projektmanagement (PM) beschreibt laut DIN69901 die *Gesamtheit von Führungsaufgaben, -organisation, -techniken und –mitteln,* die *für die Abwicklung eines Projektes* erforderlich sind.
Zugleich wird unter Projektmanagement aber auch die Institution verstanden, welche die Führung des Projektes durchführt [15]. Um Unklarheiten aufgrund der synonymen Verwendung des Projektmanagementbegriffes vorzubeugen, wird im Folgenden die Institution, welche die Führung des Projektes ausübt als Projektleitung bezeichnet.

Abbildung 2-3 zeigt die Unterteilung eines Projektes nach dem Vier-Phasen-Modell. Nachfolgend wird ausgeführt, welche Aufgaben im Rahmen der einzelnen Phasen des Projektes anfallen und welche Methoden das Projektmanagement für die Abwicklung der einzelnen Phasen bereitstellt.

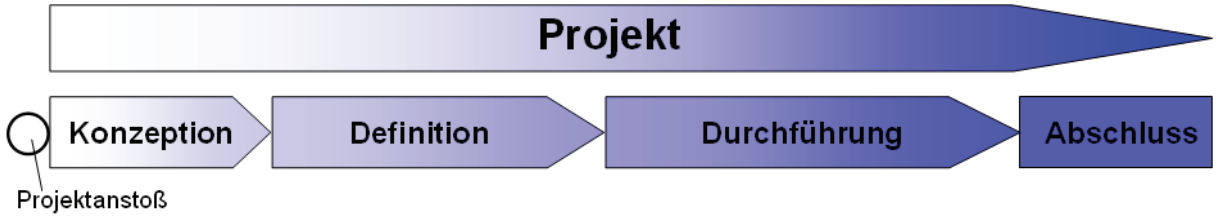


Abbildung 2-3: Vier-Phasen-Modell [17]

Projektphase 1: Konzeption
Nach dem Anstoß eines Projektes ist zunächst eine umfassende Analyse der Aufgabenstellung von Nöten. Es gilt die vom Auftraggeber vorgegebenen Termin-, Sach-, Kosten- und Nebenziele auf Eindeutigkeit, Vollständigkeit und Widerspruchsfreiheit zu prüfen. Das Ergebnis dieser Analyse ist ein Lastenheft. Dieses beschreibt die Gesamtheit aller Kundenanforderungen ohne die technische Realisierung zu detaillieren oder die Realisierbarkeit zu prüfen.
Auf Grundlage des Lastenhefts erfolgt die Erarbeitung verschiedener Lösungsvarianten, von denen schließlich ein konkreter Lösungsansatz für die Realisierung des Projektes auszuwählen ist.

Für die Konzeptionsphase sind keine Projektmanagement-Techniken erforderlich. Es empfiehlt sich stattdessen auf Erfahrungswissen aus früheren vergleichbaren Projekten zurückzugreifen oder mit Kreativtechniken zu arbeiten. Weiterhin können Machbarkeits- und Wirtschaftlichkeitsstudien die zielgerichtete Auswahl einer Lösungsvariante unterstützen.

Projektphase 2: Definition

In der Projektdefinition wird der gesamte Projektablauf festgelegt. Zu diesem Zweck wird das komplexe, unübersichtliche Gesamtprojekt in einzelne, plan- und kontrollierbare Teilaufgaben zerlegt. Nach einer einleitenden Risikoanalyse werden die Bearbeitungsreihenfolge und die Durchführungszeitpunkte der einzelnen Teilaufgaben festgelegt sowie Mitarbeiter, Maschinen und Materialien zugewiesen. Auf Grundlage dieser Informationen kann abschließend eine Kalkulation der Kosten des Projektes vollzogen werden.
Die Ergebnisse der Projektdefinition werden oftmals in einem Pflichtenheft festgehalten. Dieses Dokument detailliert die technische Realisierung des Projektes und ist vertraglich bindend. Daher wird es sowohl von den Projektverantwortlichen als auch von den Auftraggebern zu unterzeichnet.

Die Projektdefinition bildet den Kerneinsatzbereich des Projektmanagements. Daher ist für diese Phase eine Vielzahl unterschiedlicher Managementmethoden verfügbar. Aufgrund der umfassenden Thematik können im Folgenden lediglich die wichtigsten Methoden vorgestellt werden. Für weiterführende Informationen sei auf die einschlägige Literatur verwiesen, beispielsweise [15], [16] oder [18].

Abbildung 2-4 zeigt den Ablauf der Projektdefinition sowie die Hilfsmittel die in den einzelnen Arbeitsschritten eingebracht werden. Die einzelnen Elemente der Grafik werden nachfolgend erläutert.

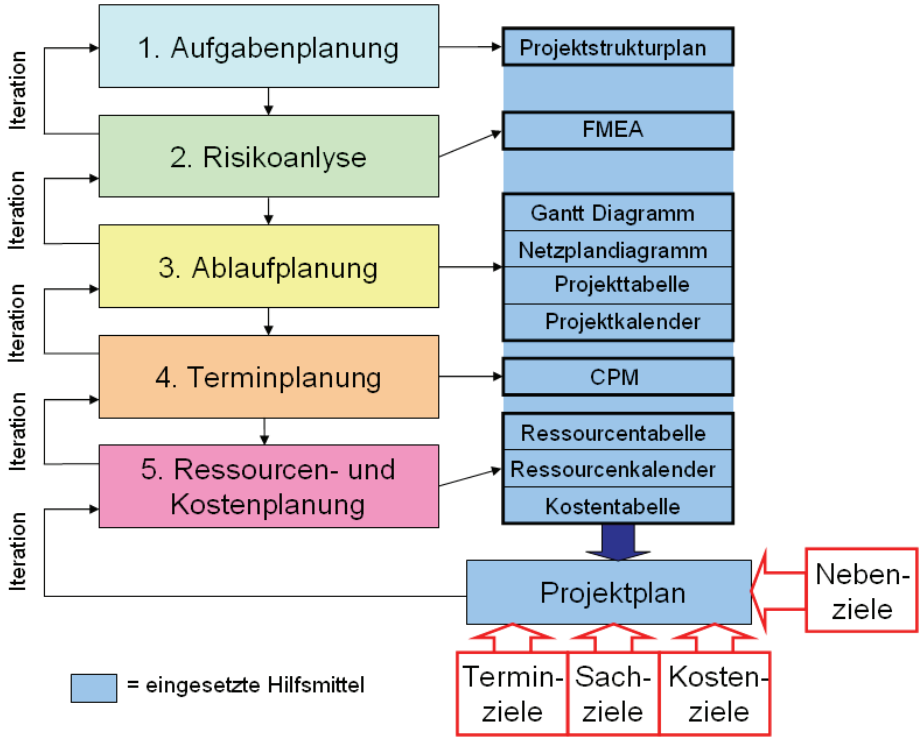


Abbildung 2-4: vereinfachter Ablauf der Projektdefinition

1. Aufgabenplanung

Das Aufbrechen des Gesamtprojektes in einzelne, plan- und kontrollierbare Arbeitsaufgaben wird mit Hilfe eines Projektstrukturplanes (PSP) dokumentiert. Dieses Planungswerkzeug ermöglicht die strukturierte Erfassung aller Teilaufgaben eines Projektes und ihrer gegenseitigen Beziehungen in Form einer Baumstruktur. Das unterste Element eines jeden Astes bildet ein sogenanntes Arbeitspaket. Dabei handelt es sich um ein abgeschlossenes, genau definiertes Arbeitsvolumen, das weitestgehend unabhängig bearbeitet werden kann. Ein Arbeitspaket kann somit vollständig an eine Arbeitsgruppe oder Abteilung vergeben werden. Die Summe aller Arbeitspakete eines Projektes umfasst folglich alle Aufgaben, die zum erreichen des Projektziels erforderlich sind.

Die Darstellung des PSPs kann grafisch (Baumstruktur oder Hierarchie), halbgrafisch (Aufgabenliste mit Einrückungen) oder tabellarisch erfolgen. Die folgende Abbildung 2-5 illustriert einen grafischen Projektstrukturplan als Beispiel.

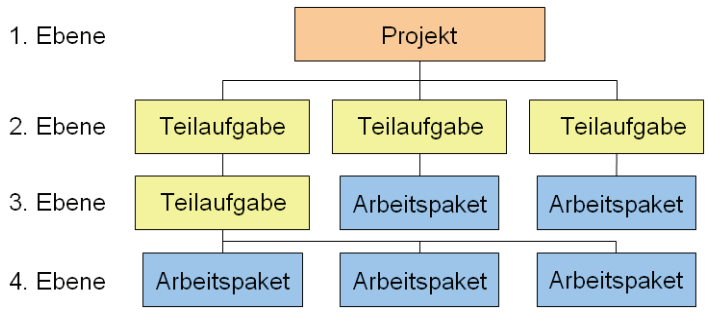


Abbildung 2-5: Projektstrukturplan [17]

2. Risikoanalyse

Projekte weisen aufgrund ihrer Komplexität eine Vielzahl unbekannter Faktoren auf, die bei der Abwicklung des Projektes zu erheblichen Schwierigkeiten führen können. Aus diesem Grund besteht eine zentrale Aufgabe des Projektmanagements darin, derartige Risiken bereits in der Definitionsphase zu erkennen, zu bewerten und durch entsprechende Maßnahmen zu vermeiden bzw. zu minimieren.

Das Risiko wird im Projektmanagement als monetäre Größe quantifiziert. Es errechnet sich aus dem Produkt der zu erwartenden Schadenshöhe und der Eintrittswahrscheinlichkeit. Im Rahmen des Projektmanagements sind vier unterschiedliche Gruppen von Risiken zu berücksichtigen:

- technische Risiken
- wirtschaftliche Risiken
- politische Risiken
- soziokulturelle Risiken

In der Fachliteratur wird eine Vielzahl von Methoden zum Auffinden von Risiken angeführt. Nachfolgend wird der Ablauf einer Risikoanalyse nach der FMEA-Methode vorgestellt. Dieses Verfahren gehört zu den erfolgreichsten Methoden zur Bewertung von Risiken und dient an dieser Stelle als repräsentatives Beispiel. Für weiterführende Informationen über die Methodik der Risikoanalyse sei auf die entsprechende Fachliteratur verwiesen (z.B. [15] oder [16]).

Die **Fehler-Möglichkeit und Einfluss-Analyse (FMEA)** ist eine Methode zur vorsorgenden Fehlerverhütung. Dabei werden zunächst die in der Aufgabenplanung definierten Arbeitspakete auf mögliche Fehlerquellen untersucht. Dabei wird im Wesentlichen jedes Arbeitspaket anhand der folgenden Fragestellungen geprüft:

- Sind bei dem Arbeitspaket qualitative Schwierigkeiten zu erwarten ?
- Sind bei dem Arbeitspaket terminliche Engpässe zu erwarten ?
- Sind bei dem Arbeitspaket finanzielle Engpässe zu erwarten ?
- Welche Auswirkungen haben bei diesem Arbeitspaket auftretende Schwierigkeiten auf andere Arbeitspakete ?

Dieser Prozess dient der Identifizierung der risikoreichen Arbeitspakete sowie der Erfassung der möglichen Schwierigkeiten. Auf Grundlage der ermittelten Informationen erfolgen schließlich eine Bewertung des Risikos und die Bestimmung einer Risikoprioritätszahl. Im

einfachsten Fall errechnet sich diese aus der Eintrittswahrscheinlichkeit und den zu erwartenden Kosten. Für eine genauere Risikoabschätzung können jedoch auch weitere Faktoren in der Risikoprioritätszahl berücksichtigt werden, z.B. zu erwartende Terminverzögerungen.

Anhand der Risikoprioritätszahl werden anschließend die Vorgänge mit den höchsten Risiken ausgewählt und gesondert betrachtet. Es erfolgt zunächst eine Analyse der Ursache der zu erwartenden Schwierigkeiten und danach die Festlegung von Maßnahmen zur Reduzierung des Risikos. Dabei ist zu unterscheiden zwischen präventiven Maßnahmen zur Vermeidung des Risikos und Überwachungsmaßnahmen sowie korrektiven Maßnahmen zur Minimierung des Risikos.

Abbildung 2-6 verdeutlicht die Quantifizierung des Risikos am Beispiels der Risikoanalyse bei der Entwicklung eines neuen Autos.

AP Nr.	Arbeitspaket	Schwierigkeiten mit großem Risiko	mögliche Ursachen	mögliche Maßnahmen	Wahrscheinlichkeit	Kosten (T€)	Prioritätszahl (T€)
110	Konstruktion Vorderachse	konstruktiver Mehraufwand	Bodenfreiheit zu groß Bauraum zu klein	Spezifikation ändern; konstruktiv geringstes Bauvolumen fordern	0,8	90	72
150	Auslegung Getriebe	Terminüberschreitung bei der Zulieferung der Auslegungsdaten	Arbeitsausfall in der Konstruktion	Überstunden anordnen; Leihkräfte einstellen	0,3	120	36
...	...	...	...	...	...	...	...

Abbildung 2-6: Risikoanalyse Autoentwicklung (Auszug) [15] [16]

3. Ablaufplanung

Die Aufgabe der Ablaufplanung besteht darin, die Bearbeitungsreihenfolge der einzelnen Arbeitspakete des Projektes festzulegen. Je nach Detaillierungsgrad des vorliegenden Projektstrukturplans ist es dabei gegebenenfalls notwendig, die in der Aufgabenplanung definierten Arbeitspakete weiter zu untergliedern.

Die Bearbeitungsreihenfolge wird mittels Vorgängen, Ereignissen und Beziehungen definiert. Ein Vorgang ist ein Geschehen, das ein eindeutiges Ziel besitzt und dessen Bewältigung Zeit erfordert. Jeder Vorgang wird durch einen definierten Anfang und ein definiertes Ende begrenzt. Diese Grenzen sind Ereignisse. Ereignisse sind als Elemente zu verstehen, die einen bestimmten Zustand kennzeichnen und im Gegensatz zu Vorgängen keine Dauer besitzen.

Das Arbeitspaket "Beschaffung" lässt sich beispielsweise in die Vorgänge "Angebote einholen", "Angebote auswerten" und "Bestellung" untergliedern. Dabei sind die Zeitpunkte an denen die einzelnen Vorgänge begonnen und abgeschlossen werden, durch Ereignisse gekennzeichnet (z.B. "Anfang Bestellung" und "Ende Bestellung").

Die Ablaufplanung erfolgt unter Anwendung der Methoden der Graphentheorie. Dadurch lassen sich die logischen und technologischen Abhängigkeiten zwischen den Vorgängen eines Projektes in Form eines Netzplans darstellen. Für die Erstellung und Abbildung eines Netzplans stellt das Projektmanagement die nachfolgenden Hilfsmittel zur Verfügung.

Das **Netzplandiagramm** besteht aus Knoten, die entweder Vorgänge oder Ereignisse repräsentieren. Die Beziehungen zwischen den einzelnen Knoten werden als Pfeile dargestellt. Abbildung 2-7 verdeutlicht, dass verschiedenartige Netzplandiagrammen unterschiedliche Sichten auf ein Projekt ermöglichen.
Im Einzelnen werden drei Arten von Netzplandiagrammen differenziert:

- Der Vorgangspfeil-Netzplan (VPN) visualisiert die Vorgänge als Pfeile und die Start- und Endereignisse der Vorgänge als Knoten.
- Der Ereignisknoten-Netzplan (EKN) bildet Ereignisse als Knoten und die Abhängigkeiten zwischen den Ereignissen als Pfeile ab.
- Beim Vorgangsknoten-Netzplan (VKN) werden die Vorgänge des Projektes als Knoten und die Anordnungsbeziehungen der Vorgänge durch Pfeile dargestellt.

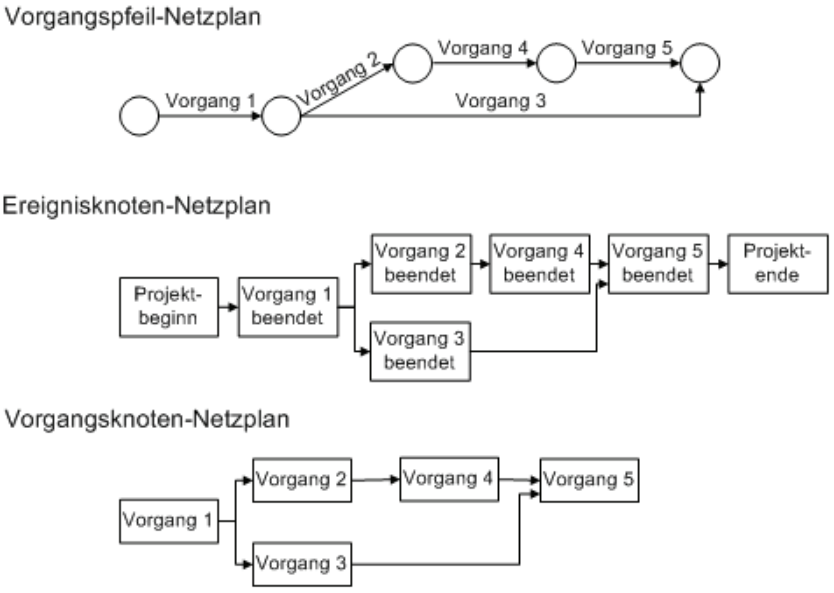


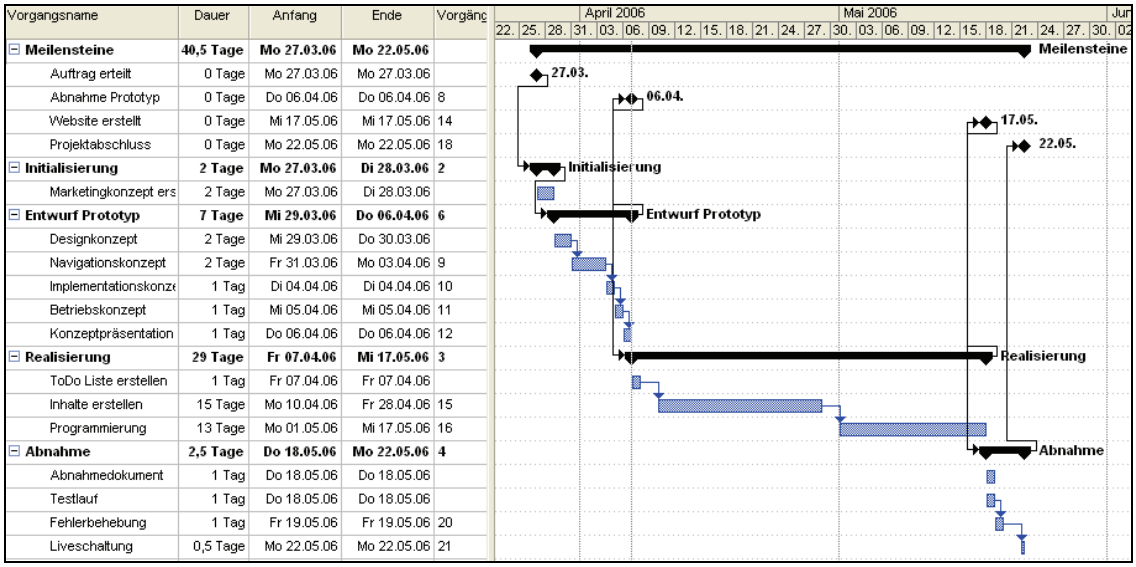
Abbildung 2-7: Netzplandiagramme [19]

Das **Gantt-Diagramm** besteht aus einer Tabelle und einem Diagramm. In der Tabelle sind Informationen zu den Vorgängen in Textform hinterlegt, während im Diagramm dieselben Informationen in Form von übereinander angeordneten Balken wiedergegeben werden. Der zu einem Vorgang gehörige Balken befindet sich dabei immer auf der selben Höhe wie die dazugehörige Tabellenzeile, während seine horizontale Lage durch eine Zeitachse bestimmt wird (vgl. Abbildung 2-8).

Die Länge des Balkens ist proportional zu seiner Dauer. Aus seiner Lage lassen sich somit sein Start- und Endtermin ablesen. Sich überschneidende Vorgänge werden durch übereinander stehende Balken visualisiert.

Weiterhin können zur Unterstützung der Grob- und Feinplanung Vorgänge zu Gruppen zusammengefasst werden sowie zur Steigerung der Transparenz Meilensteine eingefügt werden. In Abbildung 2-8 sind diese Elemente schwarz hervorgehoben.

Die Abhängigkeiten zwischen den Vorgängen werden in dem Gantt-Diagramm durch Verknüpfungsbeziehungen erfasst. Diese werden im Diagrammteil in Form von Pfeilen visualisiert. Diese Weiterentwicklung des ursprünglichen Gantt-Diagramms wird als vernetztes Gantt-Diagramm, vernetzter Balkenplan oder Plannet-Diagramm bezeichnet. Abbildung 2-8 zeigt ein vernetztes Gantt-Diagramm aus dem PM-Systems MS Project.



Vorgangsname	Dauer	Anfang	Ende	Vorgänc
Meilensteine	**40,5 Tage**	**Mo 27.03.06**	**Mo 22.05.06**	
Auftrag erteilt	0 Tage	Mo 27.03.06	Mo 27.03.06	
Abnahme Prototyp	0 Tage	Do 06.04.06	Do 06.04.06	8
Website erstellt	0 Tage	Mi 17.05.06	Mi 17.05.06	14
Projektabschluss	0 Tage	Mo 22.05.06	Mo 22.05.06	18
Initialisierung	**2 Tage**	**Mo 27.03.06**	**Di 28.03.06**	**2**
Marketingkonzept ers	2 Tage	Mo 27.03.06	Di 28.03.06	
Entwurf Prototyp	**7 Tage**	**Mi 29.03.06**	**Do 06.04.06**	**6**
Designkonzept	2 Tage	Mi 29.03.06	Do 30.03.06	
Navigationskonzept	2 Tage	Fr 31.03.06	Mo 03.04.06	9
Implementationskonze	1 Tag	Di 04.04.06	Di 04.04.06	10
Betriebskonzept	1 Tag	Mi 05.04.06	Mi 05.04.06	11
Konzeptpräsentation	1 Tag	Do 06.04.06	Do 06.04.06	12
Realisierung	**29 Tage**	**Fr 07.04.06**	**Mi 17.05.06**	**3**
ToDo Liste erstellen	1 Tag	Fr 07.04.06	Fr 07.04.06	
Inhalte erstellen	15 Tage	Mo 10.04.06	Fr 28.04.06	15
Programmierung	13 Tage	Mo 01.05.06	Mi 17.05.06	16
Abnahme	**2,5 Tage**	**Do 18.05.06**	**Mo 22.05.06**	**4**
Abnahmedokument	1 Tag	Do 18.05.06	Do 18.05.06	
Testlauf	1 Tag	Do 18.05.06	Do 18.05.06	
Fehlerbehebung	1 Tag	Fr 19.05.06	Fr 19.05.06	20
Liveschaltung	0,5 Tage	Mo 22.05.06	Mo 22.05.06	21

Abbildung 2-8: vernetztes Gantt-Diagramm (MS Project)

Neben dem Netzplan- und dem Gantt-Diagramm gibt es noch diverse Möglichkeiten den Projektablauf in Form von unterschiedlichen Tabellen und Kalenderansichten darzustellen. Da nur das Gantt- und das Netzplandiagramm einen umfassenden Überblick über das Gesamtprojekt bieten, sollen andere Ansichten an dieser Stelle nicht weiter diskutiert werden.

4. Terminplanung

Der in der Ablaufplanung erarbeitete Netzplan bildet die Grundlage für die Terminplanung. Dort werden die Durchführungszeitpunkte der einzelnen Vorgänge festgelegt und expliziten Kalenderdaten zugeordnet. Für diesen Arbeitsschritt stellt das Projektmanagement leistungsfähige Techniken zur Struktur- und Zeitplanung bereit.

Die wichtigste Methode zur Terminplanung ist die **Critical Path Method (CPM)**. Sie dient der Berechnung der Zeiten der einzelnen Vorgänge und arbeitet nach einem deterministischen Berechnungsmodell. Die Dauern der einzelnen Vorgänge müssen dabei aus Erfahrungswerten abgeschätzt werden. Aus der Vorgangsdauer und dem Netzplan erfolgen die Berechnung der zeitlichen Lage sämtlicher Vorgänge und deren Pufferzeit.

Vorgänge ohne Pufferzeit bilden Projektengpässe. Sobald einer dieser Vorgänge nicht termingerecht abgeschlossen wird, verspäten sich sämtliche nachgelagerten Vorgänge und dadurch auch die Fertigstellung des gesamten Projektes. Unabgeschlossene Vorgänge ohne Pufferzeit werden aus diesem Grund als 'kritisch' eingestuft. Eine Verkettung von kritischen Vorgängen wird als 'kritischer Pfad' bezeichnet.

Da die Fertigstellung von Projekten in der Regel von einer vergleichsweise geringen Anzahl von kritischen Vorgängen bestimmt ist, besteht ein zentraler Aspekt der Critical Path Method in der Visualisierung der kritischen Vorgänge bzw. der kritischen Pfade.

5. Ressourcen- und Kostenplanung

Im Anschluss an die Terminplanung folgt die Ressourcenplanung. Die Aufgabe der Ressourcenplanung besteht darin, die im Verlauf des Projektes benötigten Ressourcen zu erfassen und einen möglichst effizienten Einsatz der Ressourcen über das Gesamtprojekt zu erzielen. Die zentrale Ressource ist die humane Ressource, also der Mitarbeiter.

Die Ressourcenplanung kann sowohl global als auch vorgangsbasiert erfolgen. Bei der globalen Ressourcenplanung wird der Bedarf für das Gesamtprojekt geschätzt. Währenddessen wird bei der vorgangsbasierten Ressourcenplanung jedem Vorgang eine Ressource zugewiesen. Dies ermöglicht eine exakte Berechnung der Ressourcenauslastung sowie die Erfassung der Abhängigkeit zwischen der zugewiesenen Ressource und der Dauer eines Vorganges.

Für die Ressourcenplanung stellt das Projektmanagement folgende Hilfsmittel bereit: ***Ressourcentabellen*** und ***Ressourcenkalender*** dienen der gezielten Auswertung des Einsatzes einer bestimmten Ressource. Das ***Gantt-Diagramm*** ermöglicht hingegen eine Übersicht über die Ressourcenzuweisung im Kontext des Gesamtprojektes. Weiterhin können auch im ***Netzplandiagramm*** Ressourceninformationen berücksichtigt werden.

Nach der Zuteilung der Ressourcen kann eine Kostenberechnung durchgeführt werden. Dazu sind für jeden Vorgang die festen oder die variablen Kosten anzugeben. Feste Kosten sind vorgegebene Fixkosten für einen Vorgang, während die variablen Kosten sich aus den Ressourcenkosten und der Vorgangsdauer errechnen. Die Auswertung der einzelnen Kostenträger wird in der Regel in der buchhaltungsüblichen Form einer ***Kostentabelle*** vollzogen (vgl. Abbildung 2-9).

	Vorgan	Feste Kosten	Fälligkeit fester Kosten	Gesamtkosten	Geplant	Abweichung	Aktuell	Verbleibend
1	A	0,00 €	Ende	102.420,00 €	0,00 €	102.420,00 €	102.420,00 €	0,00 €
2	B	0,00 €	Anteilig	12.805,00 €	0,00 €	12.805,00 €	6.405,00 €	6.400,00 €
3	C	0,00 €	Anteilig	14.410,00 €	0,00 €	14.410,00 €	3.610,00 €	10.800,00 €
4	D	0,00 €	Anteilig	24.010,00 €	0,00 €	24.010,00 €	24.010,00 €	0,00 €
5	E	0,00 €	Anfang	43.205,00 €	0,00 €	43.205,00 €	9.605,00 €	33.600,00 €
6	F	0,00 €	Anteilig	12.805,00 €	0,00 €	12.805,00 €	9.605,00 €	3.200,00 €
7	G	0,00 €	Anteilig	86.420,00 €	0,00 €	86.420,00 €	0,00 €	86.420,00 €
8	H	0,00 €	Anteilig	9.601,00 €	0,00 €	9.601,00 €	0,00 €	9.601,00 €
9	I	0,00 €	Anteilig	24.010,00 €	0,00 €	24.010,00 €	0,00 €	24.010,00 €
10	J	0,00 €	Anteilig	1.601,00 €	0,00 €	1.601,00 €	0,00 €	1.601,00 €
11	K	0,00 €	Anteilig	6.405,00 €	0,00 €	6.405,00 €	0,00 €	6.405,00 €

Abbildung 2-9: Kostentabelle aus MS Project

Die Ergebnisse der Projektdefinition werden in Form eines Projektplanes aufbereitet. Sämtliche Vorgaben die im Laufe der Ablauf-, Termin-, Ressourcen- und Kostenplanung sowie der Risikoanalyse getätigt wurden, sind in diesem Plan dokumentiert. Wie Abbildung 2-4 verdeutlicht, ist der Planungsprozess keine Einbahnstraße. Während der Projektplan von der Ablaufplanung an stetig konkretisiert wird, erfolgt ein ständiger Abgleich mit den Projektzielen. Der Projektplan wird solange iteriert, bis er sämtlichen Zielvorgaben gerecht wird. Wenn keine entsprechende Lösung gefunden werden kann, ist eine kritische Prüfung der Projektziele erforderlich.

Die Darstellung des Projektplanes erfolgt über die in der Ablaufplanung thematisierten Ansichten, wobei nur das ***Netzplan-*** und das ***Gantt-Diagramm*** einen ganzheitlichen Überblick über das Projekt ermöglicht.

Projektphase 3: Durchführung

Die Durchführung des Projektes umfasst die Realisierung der Projektplanung und das Controlling. Hinter dem Begriff Controlling (engl. Steuerung) verbirgt sich die Verfolgung des Projektfortschrittes durch strukturierte Beschaffung und Auswertung von Daten und anschließendem Vergleich mit den Plandaten. Sobald eine gravierende Abweichung von den vorgegebenen Soll-Werten absehbar ist, helfen die Methoden des Projektmanagements bei der Ursachenforschung (Abweichungsanalyse) und bei der Auswahl geeigneter Gegenmaßnahmen.

Die Kontrolle des Projektfortschrittes wird in der Regel durch einen Soll-Ist-Vergleich realisiert. Dabei erfolgt eine Erfassung des Fortschrittes der einzelnen Vorgänge über die Angabe, zu wie viel Prozent sie zu einem gewissen Zeitpunkt abgeschlossen sind und ein Abgleich des Fortschritts mit dem vorgegebenen Projektplan.

Der Soll-Ist-Vergleich kann numerisch, halbgrafisch, grafisch oder aus einer Kombination dieser Möglichkeiten erfolgen.

- Der numerische Soll-Ist-Vergleich besteht aus dem Abgleich der Fortschritts- mit den Plandaten in entsprechenden Tabellen.
- Die halbgrafische Auswertung erfolgt über die Visualisierung des Status der einzelnen Vorgänge mittels Farben und Symbolen. Verspätete Vorgänge werden im Gantt-Diagramm zum Beispiel rot eingefärbt.
- Der grafische Soll-Ist-Vergleich erfolgt im Balkendiagramm des Gantt-Diagramms. Dort wird das Ist-Datum als vertikale Linie eingeblendet und der Fortschritt der einzelnen Vorgänge als horizontale Fortschrittslinie in dem dazugehörigen Vorgangsbalken. Sobald die Datumslinie eine Fortschrittslinie überholt, liegt der dazugehörige Vorgang hinter der Planung zurück.

Projektphase 4: Abschluss

Die Abschlussphase ist dadurch gekennzeichnet, dass nach Beendigung des Projektes die Ergebnisse präsentiert werden und eine Dokumentation aller wesentlichen Informationen erfolgt. Im Rahmen der Abschlussphase wird häufig eine kritische Reflexion des gesamten Projektes durchgeführt.

Das Projektmanagement stellt für die Abschlussphase keine gesonderten Techniken bereit. Eine vollständige Analyse des Projektes ist bereits mit den in der Projektplanung und der Projektdurchführung etablierten Methoden möglich.

2.2.2 Projektmanagement-Systeme

Aufgrund der komplexen Wirkzusammenhänge innerhalb von Projekten erfordert umfassendes, effizientes Projektmanagement den Einsatz von EDV-Unterstützung. Die Software im Umfeld des Projektmanagements reicht von Standard-Softwareanwendungen bis hin zu komplexen PM-Systemen.

Unter PM-Systemen werden speziell auf Projektmanagement-Aufgaben zugeschnittene Applikationen verstanden. Softwaretools welche die Durchführung einzelner Projektmanagement-Aufgaben unterstützen, aber kein ganzheitliches Projektmanagement ermöglichen, fallen nicht unter den Begriff Projektmanagement-System. Diese Anwendungen werden im Folgenden als Arbeitsplatzsoftware bezeichnet. Da Arbeitsplatzsoftware nicht auf die Anforderungen des Projektmanagements zugeschnitten ist, können mit ihr lediglich Teilaspekte der Projektmanagement-Thematik abgedeckt werden. Weiterhin ist bei ihnen ein wesentlicher Teil des Projektmanagements manuell durchzuführen. Da es sich bei Arbeitsplatzsoftware, wie z.B. Tabellenkalkulationsprogrammen oder Flow-Chart-Programmen, um keine Projektmanagement-Systeme im eigentlichen Sinne handelt, wird im Hinblick auf die Zielsetzung dieser Dissertation dieses Thema nicht weiter vertieft.

Projektmanagement-Systeme stellen im Wesentlichen folgende Funktionalitäten bereit:
- Planung des Projektes (Ablaufplanung, Terminplanung, Ressourcenplanung)
- Controlling des Projektes

Anwendungen die nur einzelne dieser Funktionen umfassen, sind im Bereich der Arbeitsplatzsoftware anzusiedeln.

Eine Studie über den *Stand und Trend von Softwareunterstützung für Projektmanagement-Aufgaben* der deutschen Gesellschaft für Projektmanagement ergab, dass bei den Anbietern von Projektmanagement-Software nur zwei Unternehmen als eindeutig marktrelevant einzustufen sind: Microsoft und SAP. [20]
Das PM-System Project der Firma Microsoft ist *nach rein installierten Lizenzen unangefochtener Marktführer* im Bereich der Projektmanagementsoftware. Weiterhin wird SAP *kaum als direkter Wettbewerber auf dem Markt für Projektmanagement-Software empfunden*, weil die PM-Module von SAP einem starken kaufmännischen Fokus unterworfen sind, wodurch die projektorientierte Betrachtungsweise vernachlässigt wird. [20]
Aus diesen Gründen wird die aktuelle Version von MS Project[3] vom Autor als repräsentatives PM-System erachtet. Sie wird daher im weiteren Verlauf dieser Arbeit als Referenz verwendet. Auf eine detaillierte Erläuterung der Funktionalität von MS Project muss an dieser Stelle aufgrund des umfassenden Funktionsumfangs verzichtet werden. Dazu sei auf die verfügbare Literatur verwiesen.

2.3 Kombiniertes Produktdaten- und Projektmanagement

In diesem Kapitel wird der Stand der Technik bei kombinierten PM- und PDM-Systemen behandelt. Unter diesem Begriff werden Anwendungssysteme verstanden, die neben der kompletten PDM-Funktionalität auch Methoden des Projektmanagements unterstützen.

Prinzipiell lassen sich zwei Arten von kombinierten Anwendungssystemen unterscheiden:
- PDM-Systeme die mit einem externen PM-System über eine Schnittstelle Daten austauschen
- PDM-Systeme die über ein integriertes Projektmanagement verfügen

Der erste Lösungsansatz besteht aus der Kopplung von zwei autarken EDV-Systemen, die über ein definiertes Datenformat Informationen austauschen.

[3] Microsoft Office Project Professional 2007

Kennzeichnend für eine Kopplung ist, dass die beteiligten EDV-Systeme keine einheitliche Datenstruktur besitzen. Daher müssen die Ausgabedaten des PDM-Systems in die Eingabedaten des PM-Systems umgewandelt werden und umgekehrt.
Für PDM-Systeme, die über eine offene Anwendungsarchitektur verfügen, ist theoretisch die Kombination mit jedem PM-System möglich, das über die für eine Kopplung notwendigen Schnittstellen verfügt.
Eine denkbare Kopplung auf Basis einer Dateischnittstelle wäre beispielsweise folgendes Szenario: Das PDM-System SmarTeam schreibt Informationen zu einzelnen Workflows im XML-Format aus, so dass diese danach in das PM-System MS Project eingelesen und ausgewertet werden können.

Die zweite Lösungsvariante besteht in einer Integration des Projektmanagements in ein PDM-System. Im Gegensatz zur Kopplung beruht diese Anwendungsarchitektur darauf, dass alle beteiligten Komponenten eine einheitliche Datenstruktur besitzen. Für den Datenaustausch zwischen Produktdatenmanagement und Projektmanagement ist somit keine Konvertierung erforderlich.
Als führendes kommerzielles PDM-System im Bereich der PM-Integration wird zurzeit 'Teamcenter Project' von der Firma UGS erachtet [21]. Dieses System enthält eine in die Benutzeroberfläche des PDM-Systems integrierte PM-Komponente, die *Projektteams bei der Kontrolle komplexer Zeitpläne, Kosten, Verfügbarkeiten und Ressourcen* unterstützt. Diese Funktionen beruhen auf einem integrierten PM-Modul, welches auf dem PM-System *MS Project* basiert. Da der Funktionsumfang der PM-Komponente von Teamcenter Project größtenteils mit dem von MS Project identisch ist, soll er an dieser Stelle nicht näher erörtert werden. [22]

Abschließend sei darauf hingewiesen, dass die Kombination von Produktdaten- und Projektmanagement ein relativ junges Forschungsgebiet darstellt. Zwar werden im unternehmerischen Alltag sowohl PDM- als auch PM-Systeme mit Erfolg eingesetzt, kombinierte Systeme treten bisher allerdings als klare Nischenprodukte in Erscheinung. Bislang sind nur vereinzelte Anwendungen bekannt, die Produktdaten- und Projektmanagement-Funktionalitäten vereinigen. Davon wird bisher allerdings keine Anwendung in einer erwähnenswerten Breite eingesetzt. Das kombinierte Produktdaten- und Projektmanagement nimmt in der heutigen IT-Landschaft somit nur einen vernachlässigbaren Stellenwert ein.
Der Nutzen der zurzeit verfügbaren kombinierten Systeme wird im nachfolgenden Kapitel im Zuge der Analyse der Defizite diskutiert.

2.4 Defizite der heutigen Situation

Die Defizite bei der EDV-Unterstützung des Produktentwicklungsprozesses lassen sich prinzipiell in drei Gruppen aufteilen: Defizite von PDM-Systemen, Defizite von PM-Systemen und Defizite von kombinierten Systemen.

Defizite von PDM-Systemen

PDM-Systeme haben in ihrem definierten Einsatzbereich bereits einen relativ hohen Reifegrad erreicht. Es existieren zahlreiche PDM-Systeme, welche die in Kapitel 2.2 behandelten Aspekte des Produktdatenmanagements lückenlos bereit stellen.

Der Hauptgrund für die geäußerte Kritik am heutigen Produktdatenmanagement besteht darin, dass das Management von Projekten kein integraler Bestandteil des PDM-Konzeptes ist. Das Optimierungspotenzial, welches durch eine gezielte Planung von Arbeitsaufgaben, Ressourcen und Terminen in Kombination mit einem geeigneten Controlling erzielt werden kann, wird somit durchweg vernachlässigt.

Ein weiteres Defizit besteht darin, dass in einem PDM-System keine Gesamtübersicht über ein Entwicklungsprojekt möglich ist. Zwar werden einzelne Arbeitsabläufe in Form von Prozessen (Workflows) abgebildet, ein Überblick über die Beziehungen zwischen den einzelnen Prozessen eines Projektes sowie deren zeitliche Lage, wird dagegen nicht unterstützt. Der Planungsverantwortliche ist daher nicht in der Lage schnelle Entscheidungen auf Grundlage sämtlicher relevanten Informationen zu treffen oder die Folgen einer Änderung umgehend auszuwerten.

Defizite von PM-Systemen

Heutige PM-Systeme unterstützen die in Kapitel 2.1 dargelegten Projektmanagementtechniken vollständig. Sie sind somit für die EDV-Unterstützung von Projekten, wie sie im Rahmen einer Produktentwicklung anfallen, bestens geeignet.

Das zentrale Defizit von PM-Systemen ist in der Anwenderfreundlichkeit zu sehen. Während bei PDM-Systemen aufgrund der komplexen Thematik eine vielschichtige Benutzeroberfläche und eine gewisse Einarbeitungszeit als notwendig zu erachten ist, stellt die Benutzerfreundlichkeit von PM-Systemen einen berechtigten Kritikpunkt dar.

Nahezu alle Anbieter versuchen PM-Applikationen auf dem Markt zu platzieren, die Funktionen für sämtliche denkbaren Anwendungsszenarien bereitstellen. Die Managementsysteme können somit zwar einer breiten Käuferschicht gerecht werden, es wird allerdings in der Regel nur ein Bruchteil des angebotenen Funktionsumfanges benötigt. An dieser Stelle sei ein Vergleich von Wischnewski gestattet: *Niemand würde auf den Gedanken kommen, ein Universalfahrzeug zu bauen, welches sowohl Panzer, LKW, Sportwagen und PKW*

zugleich ist. Vielmehr wird für jeden Anwendungsbereich ein spezielles Fahrzeug konzipiert und gebaut [23].

Aus der unnötigen Funktionsvielfalt von PM-Systemen resultieren mehrwöchige Schulungen und lange Einarbeitungsphasen, bis ein Anwender den Funktionsumfang des Systems sinnvoll handhaben kann. Im praktischen Arbeitsalltag müssen aufgrund von Arbeitsüberlastung und Zeitdruck, Management-Aufgaben oftmals von Mitarbeitern übernommen werden, die entweder aus Zeitgründen noch keine Schulung erhalten haben oder den Seminarstoff aufgrund von fehlender Einarbeitung und Routine lückenhaft oder gar falsch anwenden. Das Ergebnis dieser Situation ist ein lückenhaftes Projektmanagement aufgrund der mangelnden Benutzerfreundlichkeit des Systems.

Defizite von kombinierten Systemen

Wie zuvor bereits erläutert, nimmt im PDM-Konzept das Projektmanagement keinen integralen Stellenwert ein. Aus diesem Grund werden im betrieblichen Alltag Produktdatenmanagement und Projektmanagement vorrangig mit separaten Systemen durchgeführt.

Bei einem Entwicklungsprojekt wird das Projektmanagement in der Regel mit einer externen PM-Anwendung vollzogen. Die im PDM-System vorhandenen Informationen finden beim Projektmanagement somit entweder gar keine Berücksichtigung oder die Daten werden manuell zwischen den beiden Systemen übertragen. Diese manuelle Synchronisation zwischen der Projektabwicklung und dem Projektmanagement ist sowohl zeitaufwendig als auch fehleranfällig, da von keinem der beiden Systeme geprüft werden kann, ob notwendige Änderungen durchgeführt wurden.

Dieser Sachverhalt stellt eine enorme Gefährdung für die Einhaltung der Zielvorgaben sämtlicher Entwicklungsprojekte eines Unternehmens dar. Absatzschwierigkeiten, Imageverlust, Kapazitätsengpässe, Liquiditätsprobleme und letztendlich Verlust der Konkurrenzfähigkeit sind als mögliche langfristige Folgen dieser lückenhaften EDV-Unterstützung von Entwicklungsprojekten anzusehen.

Ein weiteres nicht zu unterschätzendes Problem für die Durchführung einer wirtschaftlichen Produktentwicklung besteht darin, dass dem Planungsverantwortlichen keine übersichtliche Arbeitsoberfläche zur Verfügung steht, welche die kombinierten Informationen aus dem PDM- und dem PM-System aufbereitet. Dem PDM-System fehlt die Gesamtübersicht über das Projekt. Das PM-System bildet hingegen das Gesamtprojekt ab, bietet aber keinen Zugriff auf die Projektdaten und ist daher von allen wesentlichen Informationen über das Entwicklungsprojekt abgeschnitten. Die benötigten Informationen müssen deshalb aus verschiedenen Systemen oder Modulen zusammentragen werden.

Die Projektleitung besitzt folglich keine Möglichkeit Vorgesetzten, Investoren oder Projektmitgliedern den Kontext des Projektes oder einzelner Arbeitsschritte zu veranschaulichen oder den aktuellen Stand der Arbeiten am Projekt zeitnah darzulegen.
Zudem wird dem Planungsverantwortlichen durch diese Situation das kurzfristige treffen von fundierten Entscheidungen beträchtlich erschwert. Zusätzliche Zeit- und Effizienzverluste sowie eine gesteigerte Fehlergefahr sind die unausweichliche Folge dieser Sachlage.

Die vereinzelten auf dem Markt verfügbaren Anwendungssysteme, die ein kombiniertes Produktdaten- und Projektmanagement unterstützen, können nur wenig zu einer Verbesserung dieser Situation beitragen. Daher konnten sie sich bisher in keiner Form etablieren.
Die Anwendungsarchitektur der überwiegenden Mehrheit von kombinierten Systemen beruht auf einer Kopplung von autarken PDM- und PM-Systemen. Durch entsprechende Schnittstellen können Daten zwischen den einzelnen Systemen ausgetauscht werden. Da die einzelnen Systeme über eine unterschiedliche Datenstruktur verfügen, liegt dabei lediglich eine systemtechnische Verknüpfung vor. Der Nutzen einer derart flachen Verbindung zwischen Produktdaten- und Projektmanagement ist als äußerst gering einzustufen.
Im Gegensatz dazu vertreiben vereinzelte Anbieter PDM-Systeme mit PM-Modulen. Das Potential dieses Lösungsansatzes ist aufgrund der einheitlichen Datenbasis wesentlich höher einzuordnen. Die bisher verfügbaren Anwendungssysteme bestehen allerdings lediglich aus PDM-Systemen in denen ein bestehendes PM-System eingesetzt wurde. Diese Anwendungen lassen somit ein übergreifendes Konzept für ein integriertes Produktdaten- und Projektmanagement und ein an die Anforderungen von Entwicklungsprojekten angepasstes Gesamtkonzept vermissen. Die geringe Verbreitung derartiger Systeme bezeugt, dass auch der Nutzen einer flachen Integration zwischen Produktdaten- und Projektmanagement als begrenzt anzusehen ist.

Zusammenfassend zeigt die Analyse des Stands der Technik bei Projektmanagement- und Produktdatenmanagement-Systemen, dass im Wesentlichen drei Defizite vorliegen:

D1: Effizienz- und Qualitätsverluste sowie eine hohe Fehlergefahr aufgrund des Fehlens einer umfassenden Integration zwischen Produktdatenmanagement und Projektmanagement
D2: Effizienzeinbußen und gesteigertes Fehlerrisiko infolge des Fehlens einer einheitlichen Arbeitsoberfläche, die sämtliche relevanten Informationen aus dem PDM- und dem PM-System übersichtlich visualisiert und von der aus die wesentlichen PDM- und PM-Funktionalitäten verfügbar sind
D3: Zu hoher Schulungs-, Einarbeitungs- und Bedienungsaufwand bei PM-Systemen aufgrund eines überladenen Funktionsumfanges

Die hier aufgeführten Defizite lassen sich lediglich durch die Entwicklung eines praxistauglichen Gesamtkonzeptes für eine durchgängige Integration von Produktdatenmanagement und Projektmanagement lösen. Dabei sind Bedienbarkeit und eine an die Integration angepasste Benutzeroberfläche als zentrale Randbedingungen für das zu entwickelnde Konzept zu berücksichtigen. Im folgenden Kapitel werden die Anforderungen an einen Lösungsansatz spezifiziert.

3 Anforderungen an eine Lösung

Bislang wurden Funktionsumfang und Einsatzbereiche der EDV-Systeme für das Produktdatenmanagement und das Projektmanagement beschrieben. Dabei wurde die Notwendigkeit einer Zusammenführung dieser beiden Systemtypen herausgearbeitet. In diesem Kapitel werden die fachlichen und technischen Anforderungen formuliert, die an ein System für ein integriertes Produktdaten- und Projektmanagement zu stellen sind. Dabei wird in Anlehnung an die Methodik der Softwaretechnik zwischen funktionalen und nichtfunktionalen Anforderungen unterschieden.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben, ***was*** von einem zu entwickelnden System geleistet werden soll und legen dadurch den Funktionsumfang fest. Die technische Umsetzung des Systems (Implementierung) ist dabei nicht von Belang. Es gilt vielmehr die angestrebte Wechselwirkung zwischen dem System und seiner Umwelt zu beschreiben. Dies umfasst sowohl die Interaktion mit den Anwendern als auch das Zusammenspiel mit externen Systemen.

Grundsätzlich lassen sich die komplexen funktionalen Anforderungen an ein integriertes Produktdaten- und Projektmanagement in drei Bereiche unterteilen.

Der ***erste*** Anforderungsschwerpunkt betrifft die Bereitstellung der zentralen Projektmanagementfunktionalitäten.

Der ***zweite*** Anforderungsbereich umfasst das Produktdatenmanagement und die Integration des Projektmanagements in das Produktdatenmanagement.

Der ***dritte*** Kernpunkt der Anforderungen betrifft die Benutzeroberfläche des integrierten Systems.

Im Folgenden werden diese Anforderungsbereiche einzeln ausgewertet.

3.1.1 Projektmanagement

Die Unterstützung des Projektmanagements ist eine grundlegende Voraussetzung für die Optimierung von Produktentwicklungsprojekten. Ein PDM-System mit voll integriertem Projektmanagement muss das Projektmanagement in seiner gesamten Breite unterstützen, darf aber nur soweit in die Tiefe gehen, dass jeder EDV-erfahrene Mitarbeiter alle vorliegenden Arbeitsaufgaben nach einer kurzen Einarbeitungszeit übernehmen kann. Die benötigte Projektmanagement-Funktionalität gliedert sich in die folgenden Teilbereiche:

Ablauf- und Terminplanung

• Modellierung von Netzplänen. Dies umfasst unter anderem:
  - Eingabe von Vorgängen
  - Ändern von Vorgangsattributen
  - Löschen von Vorgängen
  - Verschiebung von Vorgängen in eine andere zeitliche Lage
  - Änderung der Vorgangsreihenfolge
  - Aufsplitten von Vorgängen in mehrere Teilvorgänge
  - Vereinigung von Vorgängen
  - Modellierung von Anordnungsbeziehungen zwischen Vorgängen
  - Gruppierung von Vorgängen zur Unterstützung der Grob- und Feinplanung

• Terminplanung
  - Zuordnung von Termindaten zu Vorgängen
  - Bereitstellung von Netzplanmethoden
  - Vorgabe von Termineinschränkungen
  - Berechnung von Durchführungszeitpunkten
  - Kalendrierung[4]
  - Visualisierung von Pufferzeiten
  - Kennzeichnung von kritischen Vorgängen

Ressourcenplanung

• Administration aller Ressourcen der Firma bzw. der Organisationseinheit
• Unterstützung der projektübergreifenden Ressourcenverwaltung in einem Ressourcenpool
• Unterscheidung von Arbeitsressourcen (Mitarbeiter, Maschinen) und Materialressourcen
• Verwaltung der Arbeits- bzw. Betriebszeiten von Arbeitsressourcen in Kalendern
• Zuweisung von Ressourcen zu Vorgängen

[4] Kalendrierung: Berechnung von Kalenderdaten

- Kalkulation der Vorgangsdauern in Abhängigkeit von Kapazität und Arbeitspensum der zugewiesenen Ressourcen

Controlling

- Eingabe von Vorgangsfortschritten
- Automatischer Abgleich des Vorgangsfortschritts mit den Plandaten bezogen auf das aktuelle Datum oder ein Statusdatum
- Automatische Warnung wenn der Vorgangsfortschritt hinter der Planung zurückliegt
- Ausgabe von vordefinierten Dokumenten zum Projektstatus

Allgemeines

- Analyse der Wirkzusammenhänge zwischen den Haupteinflussgrößen eines Projektes: Bei jeder planerischen Maßnahme muss umgehend ersichtlich sein, welchen Einfluss die betreffende Aktion auf die benötigte Zeit, die erzielte Qualität und das erforderliche Geld[5] ausübt.

- Projektinformationen

  Von jedem Projekt müssen die grundlegenden Informationen erfasst werden. Dies umfasst unter anderem:
  - Name des Projektes
  - Kurzbeschreibung
  - Projektleiter
  - Kunde
  - Kommentare

3.1.2 Produktdatenmanagement und Integration

Das Produktdatenmanagement bildet die Kernkomponente eines integrierten PDM-PM-Systems. Da Fertigungsunternehmen in der Regel bereits eine an das Unternehmen angepasste PDM-Lösung besitzen, besteht die zentrale Anforderung für ein integriertes System darin das Projektmanagement unter geringem Aufwand in vorhandene PDM-Systeme einbinden zu können.

[5] Da das integrierte Projekt- und Produktdatenmanagement als Werkzeug der Produktentwicklung und nicht als Hilfsmittel für die Finanzbuchhaltung konzeptioniert wird, nimmt die Kostenüberwachung in dem erstellten Konzept einen vernachlässigbaren Stellenwert ein.

Weiterhin sind folgende Anforderungen an ein integriertes System zu stellen:

- Bei der Integration muss die komplette Funktionalität des PDM-Systems jederzeit gewährleistet bleiben (vgl. Kapitel 2.1).
- Das Projektmanagement muss im PDM-System verfügbar sein bzw. aus dem PDM-System heraus aktiviert werden können.
- Aus Kompatibilitätsgründen sollte ein nicht-integrierter Betrieb des PDM-Systems unterstützt werden. Die PM-Funktionalität muss also vollständig deaktivierbar sein.
- Die PM-Komponente benötigt direkten Zugriff auf alle für das Projektmanagement relevanten Information des PDM-Systems. Dies umfasst unter anderem:
  - sämtliche Daten zu spezifischen Projekten, Prozessen, Vorgängen und Ressourcen
  - die Metadaten zu den oben genannten Objekten
  - vom PDM-System verwaltete Dokumente
- Die PM-Komponente soll sowohl als operatives System für die Projektleitung dienen, als auch als Informationssystem für alle Projektbeteiligten.

3.1.3 Benutzeroberfläche

Eine vorrangige Aufgabe eines Systems zum integrierten Produktdaten- und Projektmanagement besteht in der Bereitstellung einer Benutzeroberfläche, welche sämtliche relevanten Informationen aufbereitet und von der aus alle wesentlichen Operationen ausgeführt werden können.

Im Einzelnen muss die Benutzeroberfläche eines integrierten Systems folgende Funktionalitäten bereitstellen:

- Konsistente und übersichtliche Abbildung sämtlicher relevanter Projektinformationen.
- Widerspruchsfreie Abbildung der Struktur des Entwicklungsprojekts sowie sämtlicher Prozesse.
- Bereitstellung der in Kap. 3.1.1 dargelegten PM-Funktionen durch entsprechende grafische Bedienelemente, Benutzerdialoge und Auswahlmenüs.

3.2 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen legen durch Definition, ***wie*** die geforderte Funktionalität zu erbringen ist, die Qualität des zu entwickelnden Systems fest. Dies wird durch Ausarbeitung eines Kataloges von Bedingungen realisiert, welche bei der Systementwicklung einzuhalten sind.

Für ein System zum integrierten Produktdaten- und Projektmanagement gelten die allgemeinen Qualitätsanforderungen, die allen EDV-Anwendungen zugrunde gelegt werden. Weiterhin bestehen spezifische Anforderungen, die sich aus der Fusion des Funktionsumfanges von PDM- und PM-Systemen ergeben.

3.2.1 Grundlegende Qualitätsanforderungen

Nach dem FURPS-Modell[6] des IEEE[7] sowie der DIN ISO 9126 und der DIN ISO 9241 bestehen folgende nicht-funktionale Anforderungen an die Qualität einer Anwendungssoftware:

Funktionalität (Functionality)

Funktionalität umschreibt das Vorhandensein von Funktionen mit festgelegten Eigenschaften. Eine Softwareanwendung muss die in den funktionalen Anforderungen beschriebene Funktionalität erfüllen und dabei Richtigkeit, Angemessenheit[8], Ordnungsmäßigkeit[9] und Interoperabilität[10] aller Funktionen gewährleisten.

Benutzerfreundlichkeit (Usability)

Die Benutzerfreundlichkeit ist Maß dafür, wie schnell sich eine Software erlernen lässt und wie einfach sich ihre Handhabung gestaltet. Die Benutzerfreundlichkeit ist maßgeblich für die Anwenderakzeptanz verantwortlich und übt somit einen enormen Einfluss auf den Produkterfolg aus. Die Benutzerfreundlichkeit wird über die Benutzeroberfläche vermittelt,

---

[6] FURPS: Qualitätsmodell welches 1985 von der Firma Hewlett-Packard entwickelt wurde und die Kategorien Funktionalität, Benutzerfreundlichkeit, Verlässlichkeit, Leistungsfähigkeit und Unterstützbarkeit unterscheidet (die englische Übersetzung dieser fünf Begriffe bildet das Akronym FURPS)

[7] Institute of Electrical and Electronics Engineers: weltweiter Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informatik [6]

[8] Angemessenheit: *Die Fähigkeit eines Software-Produkts einen geeigneten Satz an Funktionen für spezifizierte Aufgaben und Bedürfnisse des Benutzers zur Verfügung zu stellen.* [24]

[9] Ordnungsmäßigkeit: *Die Fähigkeit eines Software-Produkts anwendungsspezifische Normen oder Vereinbarungen oder gesetzliche Bestimmungen und ähnliche Vorschriften zu erfüllen.* [24]

[10] Interoperabilität: *Die Fähigkeit einer Software mit einer oder mehreren spezifizierten Komponenten zusammenzuwirken.* [24]

die im Idealfall selbstbeschreibend, individualisierbar, lernförderlich, steuerbar, aufgabenangemessen, fehlertolerant sowie erwartungskonform modelliert ist.

Verlässlichkeit (Reliability)

Die Verlässlichkeit umfasst die Teilaspekte Zuverlässigkeit, Robustheit, Stabilität und Sicherheit.

- Zuverlässigkeit umschreibt die Anforderung an ein System, die erforderlichen Funktionen in akzeptabler Zeit und mit einer akzeptablen Fehlerquote bereitzustellen.
- Robustheit ist die Fähigkeit einer Anwendung, fehlerhafte Eingaben und falsche Handhabung abfangen zu können.
- Stabilität kennzeichnet eine möglichst geringe Wahrscheinlichkeit für das Auftreten von unerwarteten Effekten, wie zum Beispiel von Programmabstürzen.
- Sicherheit beschreibt die Anforderung an ein System, ein möglichst geringes personenbezogenes Gefahrenpotential zu bieten (geringes Verletzungsrisiko) und den Schutz vor unberechtigten Zugriffen auf das System.

Leistungsfähigkeit (Performance)

Die Leistungsfähigkeit eines Systems beschreibt seine Fähigkeit ein angemessenes Leistungsniveau bereitzustellen. Die Leistungsfähigkeit eines Systems setzt sich aus einer Vielzahl von quantifizierbaren Attributen zusammen. Dies umfasst u.a. Größen wie die Latenzzeit[11], die Verfügbarkeit[12] und den Durchsatz[13].

Unterstützbarkeit (Supportability)

Die Unterstützbarkeit kennzeichnet die Möglichkeit, unter geringem Aufwand Änderungen und Erweiterungen im System durchführen zu können. Wartungsfreundlichkeit, Anpassungsfähigkeit, Übertragbarkeit (Portabilität) und Internationalisierung sind Unterpunkte der Unterstützbarkeit.

Diese grundlegenden Qualitätsanforderungen werden als wichtigste, qualitative Kriterien für die Erzielung einer zufriedenstellenden Softwarequalität angesehen. Daher muss bei der Konzeption eines integrierten PDM-PM-Systems angestrebt werden, sämtliche dieser Anforderungskriterien zu einem möglichst hohen Grad zu erfüllen. Dabei ist nachfolgend besonderes Augenmerk auf die Benutzerfreundlichkeit zu legen. Dies ist dadurch begründet,

[11] Latenzzeit: Zeitintervall zwischen einer Aktion und dem eintreten einer Reaktion

[12] Verfügbarkeit: prozentuale Kenngröße für die Einsatzbereitschaft eines Systems / Zeitraum in dem ein System für operative Zwecke zur Verfügung steht

[13] Durchsatz: Arbeit die ein System in einer bestimmten Zeitspanne bewältigen kann [25]

dass einer der Hauptmängel der verfügbaren PM-Systeme in den langen Einarbeitungsphasen und den erforderlichen Schulungen zu sehen ist (vgl. Defizite, Kap.2.4).

3.2.2 Spezifische Anforderungen an ein integriertes System

Die grundlegenden Qualitätsanforderungen sind dermaßen allgemein gehalten, dass die Formulierung weiterführender, spezifischer Anforderungen erforderlich ist, die sich aus der konkreten Anwendungsaufgabe und -umgebung ableiten.
Die spezifischen nicht-funktionalen Anforderungen, die sich für ein integriertes System ergeben, lassen sich folgendermaßen strukturieren:

Technische Anforderungen

- Allgemeingültigkeit / Anpassungsfähigkeit: Das Konzept für ein integriertes PDM-PM-System darf nicht an ein spezifisches PDM-System gebunden sein, sondern muss an beliebige PDM-Systeme anpassbar sein
- Gute Performance bei der Bearbeitung von umfangreichen Projekten
- Unterstützung des Multiprojekt- und Multiprozessmanagements
- Die Programm- und Bedienlogik muss gängigen Softwarestandards entsprechen
- Das System muss eine einfache und intuitive Benutzeroberfläche bereitstellen, um einen geringen Einarbeitungs- und Schulungsaufwand zu gewährleisten

Implementierungsanforderungen

- Logische Trennung der grundlegenden Systemfunktionen, so dass die einzelnen Elemente unabhängig voneinander implementiert und gewartet werden können
- Programmierung in etablierter Hochsprache (z.B. C++, Visual Basic, C#), um Anpassungen und Erweiterungen durch eine möglichst breite Anwenderschicht zu ermöglichen
- Die Anwendungsarchitektur muss eine schrittweise Systemeinführung unterstützen
- Ein Wechsel der Sprache des Systems muss ohne aufwendige Änderung des Quellcodes möglich sein
- Das verwendete PDM-System sollte vorzugsweise über folgende Eigenschaften verfügen:
  - modulare Anwendungsarchitektur
  - umfangreiche und gut dokumentierte Programmierschnittstelle
  - konfigurierbare Benutzeroberfläche
  - offene Systemarchitektur
  - flexible Klassenstruktur

Sicherheitsanforderungen

- Schutz vor unbefugten Zugriffen auf PDM-Dokumente
- Schutz vor unbefugten Änderungen von PDM-Dokumenten
- Beschränkung des Projektmanagements auf autorisierte Mitarbeiter

Anforderungen an die Wirtschaftlichkeit

- Geringe Hardwareanforderungen
- Geringer Produktivitätsausfall während der Einarbeitungszeit
- Geringer Schulungsaufwand
- Geringer Wartungsaufwand

3.3 Fazit

Die in diesem Kapitel vorgestellten Anforderungen dienen als Grundlage für die Ausarbeitung eines Konzeptes zum integrierten Produktdaten- und Projektmanagement. Durch eine klare Unterscheidung in funktionale und nichtfunktionale Anforderungen wurde eine deutliche Abgrenzung des Funktionsumfanges von der Implementierung etabliert. Diese Trennung ist der Findung eines kreativen Lösungskonzeptes förderlich, da bei der Ausarbeitung des benötigten Funktionsumfanges keine innovativen Ideen aufgrund der Realisierbarkeit ausgeschlossen werden. In den folgenden Kapiteln sind die ausgearbeiteten Anforderungen nunmehr auf Machbarkeit zu prüfen und in ein konkretes, realisierbares Konzept zu überführen.

4 Technische Randbedingungen

Nach der Formulierung der Anforderungen an ein integriertes Produktdaten- und Projektmanagement, ist vor der Konkretisierung eines Konzeptes und der Realisierung eines Prototyps zunächst die Auswertung der technischen Randbedingungen erforderlich.
Die Randbedingungen sind determiniert durch das verwendete Betriebssystem, den gewählten Integrationsmechanismus und die verfolgte Entwicklungsstrategie. Diese Themenbereiche werden nachfolgend erörtert.

4.1 Betriebssystem

Damit das Konzept für ein integriertes Produktdaten- und Projektmanagement für eine möglichst breite Anwenderschicht Gültigkeit besitzt, muss es an die Randbedingungen des marktführenden Betriebssystems für Arbeitsplatzrechner angepasst werden.
Microsoft Windows ist im Bereich der Betriebssysteme eindeutig als marktbeherrschend einzustufen. Kalkulationen gehen davon aus, dass die verschiedenen Windows Betriebsysteme zusammen einen Marktanteil von über 90% bei Arbeitsplatzrechnern sowie von über 50% bei Serverrechnern einnehmen. Andere Betriebssysteme, wie z.B. Linux, Unix, Mac OS oder Netware halten bei den Arbeitsplatzrechnern im Vergleich zu Windows eine vernachlässigbare Marktposition und können daher kaum als Wettbewerber angesehen werden. [6][26][27]
Die Konzeptionierung eines integrierten PDM-PM-Systems wird daher auf Grundlage des geläufigsten Windows-Betriebssystems, WindowsXP, durchgeführt. Für eine umfassende Beschreibung von WindowsXP sei auf die Produktdokumentation von Microsoft sowie die entsprechende Fachliteratur verwiesen.

4.2 Integration

Ein zentraler Aspekt dieser Arbeit besteht in der Integration von PDM- und PM-Systemen. Aus diesem Grund werden in diesem Kapitel Integrationsmechanismen thematisiert. Einleitend soll zunächst eine Abgrenzung des Begriffs 'Integration' von dem verwandten Terminus 'Kopplung' erfolgen.

Kopplung und Integration bilden zwei verschiedene Ansätze für die Vernetzung von getrennten EDV-Systemen. Eine **Kopplung** zeichnet sich dadurch aus, dass die zu verbindenden EDV-Systeme über unterschiedliche Datenstrukturen verfügen. Die Vernetzung der Systeme wird durch eine datentechnische Verknüpfung realisiert. Dabei wird das Ausgabeformat von

System A in das Eingabeformat von System B umgewandelt und umgekehrt. Somit beträgt die Anzahl von benötigten Verknüpfungen für n beteiligte Systeme: n * (n-1).
Bei einer **Integration** besitzen alle beteiligten EDV-Systeme eine einheitliche Datenstruktur. Dadurch können die einzelnen Systeme ohne Datenkonvertierung miteinander kommunizieren. Bei einer Integration werden daher deutlich weniger Verknüpfungen als bei einer Kopplung benötigt. Die Anzahl der Verknüpfungen ist abhängig von der gewählten Integrationsstrategie. Bei einer Integration lassen sich die nachfolgend aufgeführten Strategien unterschieden. [14]

Integration durch menschlichen Eingriff

Hierbei werden Informationen zwischen den integrierten Systemen durch den Menschen übertragen. Dabei handelt es sich um keine Integration im EDV-technischen Sinne, sondern um eine organisatorische Verbindung zwischen zwei autonomen Systemen. Wie in der Einleitung bereits ausgeführt wurde, ist diese Art der Integration aufgrund der hohen Fehlergefahr und des immensen Aufwandes als Lösungsansatz gänzlich ungeeignet.

Integration durch formatierte Datei

Bei diesem Verfahren werden Daten zwischen den integrierten Systemen mittels Dateien in einem vordefinierten Format transportiert. Alle beteiligten Systeme müssen dabei das verwendete Format kennen. In der Regel werden für derartige Integrationen tabulatorgetrennte, zeichengetrennte oder XML[14]-basierte Dateiformate oder binäre Formate verwendet.

Integration durch Server

Integrationsroutinen können auf einem Server ausgelagert werden. Das Client-System greift auf Funktionen des Servers zu, mit deren Hilfe die Daten an das Zielsystem übertragen werden.

Integration durch Datenbanken

Dieser Lösungsansatz besteht aus dem Austausch von Daten über ein Datenbanksystem. Dabei kann entweder eine zentrale Datenbank für alle Systemdaten verwendet werden oder es wird auf eine Datenbank zurückgegriffen, die lediglich Datenbestände verwaltet, welche für die Kommunikation erforderlich sind.

[14] XML (Extensible Markup Language): Sprache zur *Darstellung hierarchischer Strukturen in Form von Textdateien* [6]

4.3 Softwareentwicklung

Objektorientierte Softwareentwicklung (OOSE) stellt den Stand der Technik bei der Entwicklung von Softwareanwendungen dar. Nachfolgend werden zunächst die grundlegenden Begriffe der objektorientierten Programmierung vorgestellt und die Vorgehensweise bei der objektorientierten Softwareentwicklung diskutiert. Anschließend erfolgt eine Analyse möglicher Entwicklungsstrategien.

4.3.1 Begriffe

Objekt

Wie die Begriffe ‚objektorientierte Programmierung' und ‚objektorientierte Softwareentwicklung' nahe legen, steht bei diesen Verfahren der Objektbegriff im Mittelpunkt.

Objekte sind abgeschlossene Einheiten, die sich durch charakteristische Eigenschaften und spezifisches Verhalten auszeichnen.

Der Zustand eines Objektes wird anhand der Werte von Objektvariablen (Attributen) definiert, während das Verhalten eines Objektes durch die möglichen Operationen (Methoden) bestimmt wird. Abfragen und Änderungen des Zustandes bzw. der Attributswerte eines Objekts sind laut der Theorie der Objektorientierung nur mittels entsprechender Methoden des Objektes möglich. Somit werden bei der objektorientierten Programmierung Daten und Funktionen miteinander verbunden (vgl. Kapselung).

Klasse

Alle gleichartigen Objekte, d.h. Objekte mit gleichen Attributen und identischen Methoden, gehören zu einer gemeinsamen Klasse. Eine Klasse definiert welche Attribute und Methoden ein Objekt besitzt und kann daher als Bauplan für die Erzeugung von Objekten bezeichnet werden. Von einer Klasse können beliebig viele Objekte erzeugt werden, während jedes erzeugte Objekt genau einer Klasse angehört. Ein Objekt wird daher als Instanz seiner Klasse bezeichnet.

Abbildung 4-1 veranschaulicht das Konzept von Klassen und Objekten anhand der Instanzierung[15] von zwei Objekten der Klasse Person. Als Darstellungsform wird dabei die so genannte Donut-Ansicht verwendet. Diese bildet die Attribute von Klassen bzw. die Attributwerte von Objekten in einem Kreis ab, um den die zur Verfügung stehenden Methoden in einem Ring dargestellt werden.

[15] Instanzierung: Bildung einer Instanz

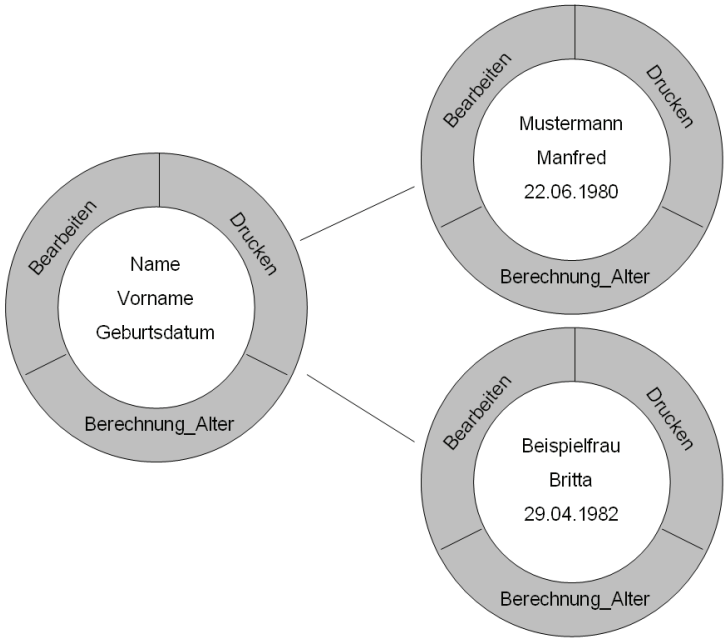


Abbildung 4-1: Objekte einer Klasse

Die Klasse Person beinhaltet die Attribute Name, Vorname und Geburtsdatum und stellt die Methoden Bearbeiten, Drucken und Berechnung_Alter zur Verfügung. Aus dieser Klasse wurden zwei Objekte instanziert, die verschiedene Attributswerte angenommen haben (Name = Mustermann, Vorname = Manfred, usw.). Beiden Objekten stehen die in der Klasse definierten Methoden zur Verfügung.

Objektidentität

Die Attribute von Objekten einer Klasse können die gleichen Werte annehmen. Die einzige Ausnahme stellt die Objektidentität dar. Dieses besondere Attribut dient der eindeutigen Identifizierung von Objekten. Daher kann ein bestimmter Identitätswert nur von jeweils einem Objekt angenommen werden. Weiterhin ist die Änderung der Identität nicht möglich.
Bei der Erfassung des Studentenbestandes einer Universität könnte es zum Beispiel durchaus vorkommen, dass zwei Objekte der Klasse Student dieselben Werte bei den Attributen ‚Name' und ‚Vorname' annehmen. Anhand der Objektidentität ‚Matrikelnummer' sind sie jedoch eindeutig unterscheidbar.

Vererbung

Mittels Vererbung können bei der Definition von neuen Klassen die Attribute und Methoden von bereits vorhandenen Klassen übernommen werden. Die Vererbung ist daher Grundlage für die Widerverwendbarkeit von Softwareteilen sowie die Etablierung von Klassenhierarchien. Die Klasse die Eigenschaften weitergibt wird als Basisklasse oder Originalklasse bezeichnet, während die Klasse die Eigenschaften erbt Subklasse oder Tochterklasse

genannt wird. Weiterhin wird zwischen Klassen unterschieden von denen Objekte erzeugt werden können (konkrete Klassen) und solchen, die lediglich zur Zusammenfassung von Eigenschaften dienen, die zur Etablierung einer Klassenhierarchie erforderlich sind. Da von diesen Klassen keine Objekte abgleitet werden können, werden sie als abstrakte Klassen bezeichnet.

Abbildung 4-2 verdeutlicht diese Zusammenhänge anhand der Vererbungsbeziehungen zwischen der abstrakten Basisklasse Uhr und den konkreten Tochterklassen Digitaluhr und Analoguhr. Die Tochterklassen erben sämtliche Attribute und Methoden der Basisklasse Uhr. Weiterhin werden die Tochterklassen um klassenspezifische Merkmale erweitert. Während bei der Klasse Digitaluhr beispielsweise das Display ein spezifisches Attribut darstellt, wird die Klasse Analoguhr um die Merkmale Stunden- und Minutenzeiger erweitert. Der Vorteil der Vererbung gegenüber von mehreren Neudefinitionen besteht in diesem konkreten Beispiel darin, dass die gemeinsamen Attribute und Methoden der Klassen Digital- und Analoguhr lediglich einmal in der Basisklasse Uhr implementiert werden müssen. Zudem werden dadurch eventuelle Änderungen an der Basisklasse dynamisch in die Tochterklassen übernommen.

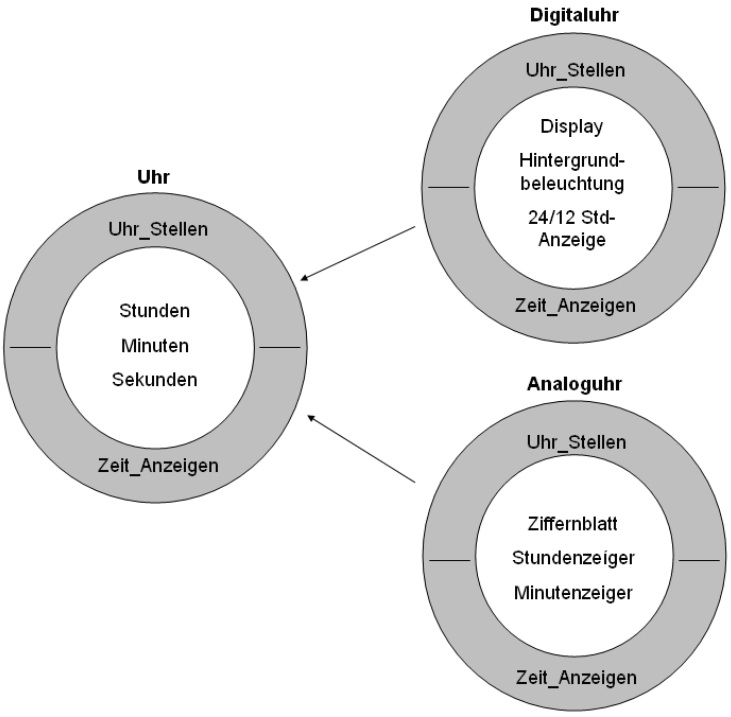


Abbildung 4-2: Vererbung

Dabei ist die Vererbung nicht auf eine einzelne Basisklasse beschränkt. Viele Programmiersprachen unterstützen das Prinzip der Mehrfachvererbung, mit der Attribute und Methoden von verschiedenen Klassen übernommen werden können. Dabei ist allerdings zu beachten, dass keine Konflikte aufgrund von gleichnamigen Attributen oder Methoden in den

Basisklassen auftreten. Durch Anwendung der Mehrfachvererbung könnte aus den Klassen Digital- und Analoguhr beispielsweise eine Uhr mit kombinierter Digital- und Analoganzeige abgleitet werden.
Weiterhin können in einer Tochterklasse die geerbten Eigenschaften auch verändert werden. Dies wird als ‚überschreiben' der Basisklassen bezeichnet.

Für gebräuchliche Programmieraufgaben, wie z.B. die Implementierung von Datenstrukturen oder die Einrichtung einer grafischen Benutzeroberfläche, sind in Entwicklungsumgebungen meist eine Auswahl von vordefinierten Klassen(bibliotheken) vorhanden. Diese können entweder unverändert in das Programm eingebunden werden oder mittels Vererbung an die vorliegenden Gegebenheiten angepasst werden.

Kapselung

Ein grundlegendes Prinzip der objektorientierten Programmierung ist die Kapselung. Die Kapselung erzielt eine Abschottung der Klassen, indem die Änderung von Attributwerten sowie deren Ausgabe ausschließlich über die bereitgestellten Klassenmethoden erlaubt wird. Von außerhalb einer Klasse ist die interne Funktionsweise folglich nicht ersichtlich. Nachdem eine Klasse einmal definiert ist, fungiert sie somit als eigenständige Einheit, die bestimmte Funktionen erfüllt ohne dass Implementierungsdetails von Bedeutung sind.

Nachricht

Die objektorientierte Programmierung beruht im Gegensatz zu der prozeduralen Programmierung nicht auf einem sequentiellen Hauptprogramm, sondern besteht aus einem Netzwerk von Objekten. Die einzelnen Objekte kommunizieren durch den Austausch von Nachrichten miteinander. Sie senden durch Aufruf ihrer Methoden Nachrichten an andere Objekte und reagieren auf eingehende Nachrichten durch Ausführung von Methoden. Dieses Senden und Empfangen von Nachrichten stellt im Sinne der Kapselung die einzige Schnittstelle der Objekte zur Außenwelt dar.

Polymorphie

Die Polymorphie[16] bildet neben Vererbung und Kapselung die dritte Säule der objektorientierten Programmierung. Die Polymorphie ermöglicht es bei unterschiedlichen Klassen für gleichartige Methoden denselben Namen zu verwenden. Da die Objekte aufgrund der Polymorphie auf Nachrichten klassenspezifisch reagieren, muss der Sender einer Nachricht gar nicht wissen zu welcher Klasse ein Objekt gehört. Das Objekt führt automatisch die richtige Methode aus.

---

[16] Polymorphie: <griech. polýmorphos> Vielgestaltigkeit, viele Erscheinungsformen [12]

Assoziation, Aggregation und Komposition

Assoziationen sind logische Beziehungen zwischen einzelnen Klassen, aus denen sich Verknüpfungen zwischen den aus diesen Klassen erzeugten Objekten ergeben. Dadurch lassen sich funktionale Zusammenhänge[17] und Datenabhängigkeiten[18] zwischen den Objekten von assoziierten Klassen modellieren.

Bei Aggregation und Komposition handelt es sich um Spezialfälle der Assoziation, bei denen zwischen den Objekten von Klassen eine Rangordnung besteht, weil eine Klasse ein Element einer anderen Klasse ist. Abbildung 4-3 illustriert dies am Beispiel der Klasse Artikel, die ein Element der Klasse Katalog ist.

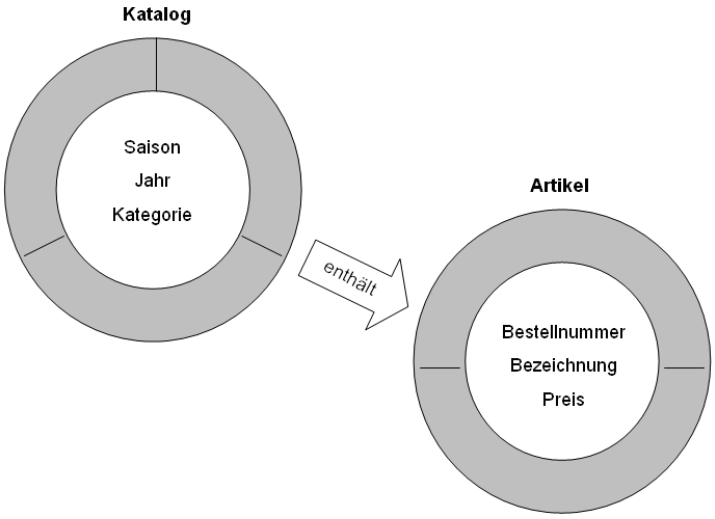


Abbildung 4-3: Datenabhängigkeit und Aggregation [28]

Eine Aggregation liegt vor, wenn ein Objekt einer Klasse nach der Auflösung der verknüpften Klasse weiter existiert, während die Komposition dadurch gekennzeichnet ist, dass eine Auflösung des ranghöheren Objektes auch eine Auflösung des rangniedrigeren Objektes nach sich zieht. Bei Abbildung 4-3 handelt es sich beispielsweise um eine Aggregation. Ein Objekt der rangniedrigeren Klasse Artikel kann nach Auflösen eines Objektes der Klasse Katalog alleine weiter existieren und zum Beispiel im Katalog des Folgejahres verwendet werden.

[17] Funktionaler Zusammenhang: Ein Objekt benötigt die Methoden eines anderen Objektes, um die eigene Funktionalität zu gewährleisten.

[18] Datenabhängigkeit: Ein Objekt benötigt Informationen über die Attribute eines anderen Objektes oder ist Teil der Attribute eines anderen Objektes

4.3.2 Entwicklungstechnologien

Die Hauptaufgabe moderner Software-Entwicklungstechnologien besteht in der Unterstützung der modularen Softwareentwicklung. Grundvoraussetzung für die Modularisierung von Computerprogrammen ist die Verwendung einer objektorientierten Programmiersprache. Diese ermöglicht durch das Prinzip der Kapselung die Untergliederung des Programmcodes in funktionale Einheiten, die Klassen. Mittels des objektorientierten Prinzips der Vererbung wird gewährleistet, dass einmal definierte Klassen beliebig wieder verwendet werden können. Sie können somit in mehreren Programmen bzw. Programmteilen eingesetzt werden.
Im Umfeld der objektorientierten Programmierung haben sich folgende Technologien etabliert, die den Programmierer bei der modularisierten Softwareentwicklung unterstützen:

Programmschnittstelle (API [19])

Eine Programmschnittstelle ist eine dokumentierte Softwareschnittstelle, mit der Programmierer auf Funktionen einer anderen Anwendung oder eines Betriebssystems zugreifen können. Zu diesem Zweck stellt jede API einen Satz an Funktionsaufrufen bereit, mit denen in ein Programm Funktionen einer anderen Anwendung eingebunden werden können. Der Zugriff auf Programm- und Klassenbibliotheken wird gleichermaßen über API-Aufrufe realisiert.

Klassenbibliotheken

Klassenbibliotheken sind Sammlungen von anwendungsneutralen Klassen, die zum Zweck der Wiederverwendung ausgelagert sind. Bibliotheksklassen können auf zweierlei Arten in einen Programmentwurf eingebunden werden:

- Instanzierung von Objekten aus den Bibliotheksklassen
- Verwendung der Bibliotheksklassen als Basisklassen für die Implementierung von neuen Tochterklassen

Framework

Frameworks[20] stellen einen Sonderfall von Klassenbibliotheken dar. Die Klassen einer normalen Bibliothek sind in keinem übergreifenden Gesamtzusammenhang zu sehen. Ein Framework stellt hingegen kooperierende Klassen zur Verfügung, die als Bausteine eines wiederverwendbaren Entwurfs für eine bestimmte Art von Software zu verstehen sind. Das Framework kann vom Programmierer für eine konkrete Anwendung angepasst werden, indem er Tochterklassen der Frameworkklassen bildet. Frameworks ermöglichen somit nicht

[19] API: <Abk.> Application Programm Interface

[20] Framework: <engl.> Rahmenwerk, (Programm-)Gerüst

nur die Wiederverwendung von Programmcode, sondern auch die des grundlegenden Programmdesigns.

Templatebibliotheken

Templates[21] sind Programmelemente, die einen vom Datentyp unabhängigen Quellcode besitzen. Dies erlaubt die Implementierung von Klassen und Funktionen, die in der Lage sind Argumente von verschiedenen Datentypen entgegenzunehmen und diese in derselben Weise zu verarbeiten. Technisch werden Templates durch die Anwendung der generischen Programmierung realisiert. Dabei werden die Datentypen der Programmvariablen nicht spezifiziert, sondern mit Platzhaltern belegt. Dadurch werden Definition und Deklaration von Klassen- und Funktionstemplates während der Programmierung variabel gehalten und erst beim Kompilieren bestimmt. Voraussetzung für die Verwendung von Templates ist, dass für den eingesetzten Datentyp sämtliche in den Templates implementierten Operatoren definiert sind. Dies wird bei selbstdefinierten Datentypen durch die Überladung der Operatoren erzielt (vgl. Prinzip der Polymorphie). Die Templatetechnologie ermöglicht dadurch die Erstellung von Vorlagen für gebräuchliche Funktionen und Klassen und deren Wiederverwendung unabhängig vom Datentyp. Analog zu Klassen sind auch für Templates eine Vielzahl von verschiedenen Bibliotheken verfügbar, in denen ein breites Spektrum an vorgefertigten Templates hinterlegt ist.

Dynamische Laufzeitbibliotheken

Bei dynamischen Laufzeitbibliotheken handelt es sich um Programmbibliotheken, die wesentliche Softwarefunktionen bereitstellen. Allgemeingültige Funktionen müssen somit nicht im Quellcode implementiert werden, sondern werden erst während der Ausführung des Programms (Laufzeit) aus der entsprechenden Bibliothek eingebunden. Im Gegensatz zu statischen Bibliotheken werden Elemente einer dynamischen Laufzeitbibliothek erst bei Bedarf in den Speicher geladen und dynamisch mit dem Programm verbunden. Dadurch muss der Programmcode von dynamischen Laufzeitbibliotheken, unabhängig davon wie viele Anwendungen zeitgleich dieselbe Bibliothek verwenden, nur einmal geladen werden. Daraus resultiert eine erhebliche Reduzierung des benötigten Speicherbedarfes. Weiterhin fördert der Einsatz von dynamischen Laufzeitbibliotheken die Modularität. Da in der Regel mehrere Anwendungen dieselben dynamischen Laufzeitbibliotheken verwenden, muss bei einer Systemaktualisierung nicht der Programmcode der einzelnen Anwendungen, sondern nur die entsprechende Bibliothek bearbeitet werden.

[21] Template: <engl.> Schablone, Vorlage

4.3.3 Entwicklungsstrategien

Für die Entwicklung von Windows-Anwendungen sind zurzeit zwei unterschiedliche Strategien gebräuchlich: Die objektorientierte, betriebssystemnahe Programmierung und die .net-Programmierung. Nachfolgend werden diese beiden Entwicklungsstrategien gegenübergestellt.

Objektorientierte, betriebssystemnahe Programmierung

Die OOB-Programmierung[22] folgt mit Objektorientierung und Betriebssystemnähe zwei grundlegenden Entwicklungsmaximen. Die Anwendung der Objektorientierung ermöglicht eine modulare Softwareentwicklung, worauf aufbauend eine auf das Ziel-Betriebssystem abgestimmte, betriebssystemnahe Programmierung erfolgt. Diese wird durch die Verwendung entsprechender Klassenbibliotheken, Frameworks und Templatebibliotheken realisiert, welche ein breites Spektrum von Programmfunktionen auf Grundlage der Betriebssystem-API bereitstellen. Das Ergebnis der OOB-Programmierung sind Anwendungen, die aus Programmen in kompiliertem Maschinencode bestehen.

Im Windows-Umfeld steht zur Unterstützung der OOB-Programmierung eine Vielzahl unterschiedlicher Technologien zur Verfügung. Da Anwendungsprogrammierung nicht im Vordergrund dieser Arbeit steht, erfolgt im Anschluss lediglich eine Kurzvorstellung der für eine Integration relevanten OOB-Technologien. Weiterführende Informationen sind der Fachliteratur zu entnehmen (z.B. [29] oder [30]).

- Win32-API
  Primäre Programmierschnittstelle des Betriebssystems Windows XP.

- Microsoft Foundation Class (MFC)
  Framework für die Anwendungsprogrammierung unter Windows. Es bietet eine Grundstruktur für den Programmaufbau und stellt eine große Anzahl von Klassen für eine Vielzahl von Anwendungsfällen zur Verfügung.

- Active Template Library (ATL)
  Klassenbibliothek mit großer Auswahl an vorgefertigten Templates für die Programmiersprache Visual C++.

- C++ Standardbibliothek
  C++ Bibliothek die häufig benötigte Programmelemente in einer standardisierten Form bereitstellt.

[22] OOB-Programmierung: im Rahmen dieser Arbeit verwendete Abkürzung für objektorientierte, betriebssystemnahe Programmierung

• Common Object Model (COM)
  Standard zur Erstellung von universell einsetzbaren Softwarekomponenten, der eine versionsunabhängige und plattformunabhängige Programmierung ermöglicht.

Die OOB-Programmierung stellt eine bewährte Entwicklungstechnologie bereit, die durch eine jahrelange Weiterentwicklung ständig optimiert wurde und deren Konzept sich bereits bei der Entwicklung unzähliger Anwendungen bewährt hat. Es zeichnet sich allerdings ab, dass die OOB-Programmierung in einigen Bereichen an konzeptbedingte Grenzen stößt. Die Installation von neuen OOB-Programmen kann beispielsweise bestehende Installationen beeinträchtigen, da aufgrund einer fehlenden Versionierung bestehende DLLs[23] gleichen Namens einfach überschrieben werden („Dll-Hell"). Weitere Kritikpunkte bilden die COM-Technologie und die Win32-API. Während die COM-Technologie aufgrund ihrer Komplexität als schwer handhabbar gilt, wird die Win32-API als inkonsistent und unübersichtlich angesehen. Diese Faktoren haben in Kombination mit der wachsenden Nachfrage an webbasierten und plattformunabhängigen Anwendungen Microsoft dazu veranlasst mit der .net-Programmierung eine Alternative zur OOB-Programmierung zu entwickeln. Um einen Vergleich der beiden unterschiedlichen Entwicklungsstrategien zu ermöglichen, wird nachfolgend die .net-Programmierung vorgestellt.

.net-Programmierung

Seit 2002 wird von Microsoft mit der .net-Programmierung eine zweite Entwicklungsstrategie für Windows-Applikationen unterstützt. Die .net-Plattform bietet eine Umgebung zur Programmierung und Ausführung von objektorientierten Anwendungen, die sich durch Programmiersprachenneutralität und Betriebssystemunabhängigkeit auszeichnet. Technisch wird dies durch die Verwendung einer virtuellen Laufzeitumgebung[24] realisiert, die als Middleware zwischen den .net-Programmen und dem Betriebssystem agiert. Das Ergebnis der .net-Programmierung sind Anwendungen, die aus Programmen in Bytecode bestehen und erst zur Laufzeit in Maschinencode übersetzt werden.

Das .net-Framework steht im Zentrum der .net-Technologie. Es besteht aus der Base Class Library (BCL) und der Common Language Runtime (CLR). Bei der **Base Class Library**

[23] Dynamic Link Library (DLL): Im Windows Umfeld verwendete dynamische Laufzeitbibliothek. DLLs stellen einen wesentlichen Teil der elementaren Funktionen des Windows-Betriebssystem und unzähliger Windows-Anwendungen bereit.

[24] Laufzeitumgebung (engl. Runtime): Softwareschicht die zwischen dem Betriebssystem und den Anwendungsprogammen eingefügt ist, um den Betrieb von plattformunabhängigen Applikationen zu ermöglichen.

handelt es sich um eine objektorientierte Klassenbibliothek, mit der alle Aspekte der Windows- und Webprogammierung abgedeckt werden. Die **Common Language Runtime** fungiert als virtuelle Laufzeitumgebung, welche die Ausführung der entwickelten Programme abwickelt.

Die .net-Programmierung bietet das Potenzial die Programmierung von Anwendungen grundlegend zu vereinfachen und die erstellten Anwendungen entscheidend zu verbessern. Bei der Konzeption der .net-Programmierung wurden die wesentlichen Kritikpunkte der OOB-Programmierung adressiert und die stark wachsenden Anwendungsbereiche Internet und Datenaustausch fokussiert. Weiterhin bietet die .net-Programmierung zusätzliche Verbesserungen wie Sprachneutralität und Plattformunabhängigkeit.

Eine differenzierte Betrachtung der .net-Technologie offenbart jedoch einige Kritikpunkte. So sind beispielsweise alle Anwendungen von der Common Language Runtime abhängig. Der praktische Umgang mit .net-Anwendungen zeigt, dass die Common Language Runtime keineswegs fehlerfrei arbeitet [26]. Daher stellt sich die Frage, wann und in welcher Form die Common Language Runtime aktualisiert wird und ob danach alle bisherigen .net-Anwendungen weiterhin auf ihr lauffähig sind.

Ferner ist die Quellcodekompatibilität der unterschiedlichen Programmiersprachen innerhalb von .net nicht vollständig gegeben. Die Array-Grenzen von Visual Basic sind beispielsweise inkompatibel gegenüber anderen .net-Programmiersprachen wie z.B. Visual C++ oder Visual C#.

Das größte Defizit der .net-Programmierung zeigt sich allerdings erst im direkten Vergleich mit der OOB-Programmierung. Die CLR-Laufzeitumgebung kann Programmcode nicht annähernd so schnell ausführen wie ein echter Prozessor und braucht zudem einen großen Overhead[25] an Hauptspeicher. Abbildung 4-4 verdeutlicht diesen Effekt anhand eines Vergleiches des Hauptspeicherverbrauches zweier Anwendungen, die jeweils ein Dialogfeld mit "Hello World" ausgeben. Es ist ersichtlich, dass die .net-Anwendung mehr als sieben Mal soviel Hauptspeicher verbraucht wie die entsprechende OOB-Anwendung[26] [31].

Zusammenfassend sind sowohl die OOB- als auch die .net-Programmierung zweifelsfrei innovative und leistungsfähige Entwicklungsstrategien. Zurzeit dominieren OOB-basierte Anwendungen das Microsoft-Umfeld. Dies ist darauf zurückzuführen, dass die .net-Technologie erst vor wenigen Jahren eingeführt wurde und die Umstellung auf eine neue Entwicklungsstrategie einen langwierigen Prozess darstellt.

---

[25] Overhead: <engl.> Verwaltungsdaten; Daten die nicht zu den Nutzdaten zählen, sondern als Zusatzinformationen benötigt werden [6]

[26] Bei umfangreichen Anwendungen verändert sich dieses Verhältnis zugunsten der .net-Anwendung.

Welche der beiden vorgestellten Entwicklungsstrategien zu bevorzugen ist, hängt im Wesentlichen von der zu entwickelnden Anwendung sowie deren Einsatzbereich ab. Während der Schwerpunkt der .net-Technologie in der Erstellung von verteilten Anwendungen und Webanwendungen zu sehen ist, nimmt die OOB-Programmierung im Bereich von Windows-Anwendungen einen berechtigten Stellenwert ein. Besonders im Hinblick auf Laufzeiteffizienz und Hauptspeicherverbrauch bieten die vorkompilierten OOB-Anwendungen wesentliche Vorteile gegenüber den zur Laufzeit übersetzten .net-Anwendungen.

Aufgrund der für ein integriertes Produktdaten- und Projektmanagement benötigten Performance-Eigenschaften erfolgt die Konzeptionierung eines integrierten Gesamtsystems auf Grundlage der OOB-Programmierung.

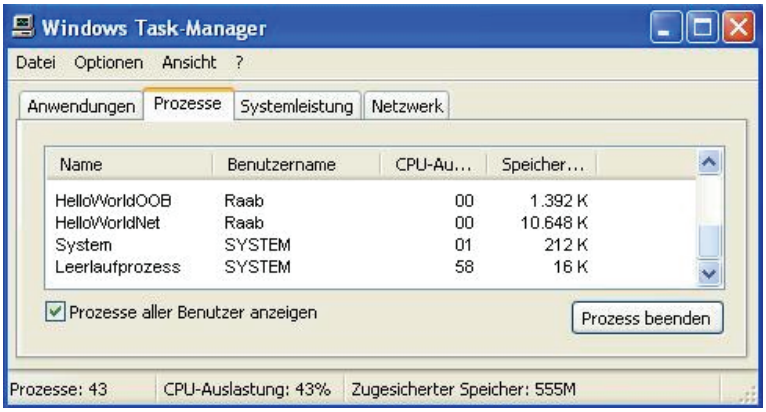


Abbildung 4-4: Vergleich des Speicherverbrauchs zwischen .net- und OOB [31]

4.4 Programmiersprachen

Da die Entwicklung des integrierten PDM-PM-Systems unter Anwendung der OOB-Programmierung erfolgt, wird die Realisierung des Konzeptes mit der Programmiersprache C++ durchgeführt. Bei C++ handelt es sich um eine objektorientierte Programmiersprache, die von den OOB-Technologien vollständig unterstützt wird. Neben einer sehr guten Performance spricht vor allem eine hohe Verfügbarkeit für diese Sprache. Im technischen Umfeld ist C++ als Quasi-Standard etabliert, wodurch fast alle EDV-Systeme eine C/C++ API bieten. Für eine Beschreibung des Sprachumfangs von C++ wird auf die einschlägige Fachliteratur verwiesen.

Neben der Anwendungsprogrammierung sind im Rahmen der Konzeptionierung Anpassungen an eingebundener Standardsoftware nötig, die auf Grund technischer Randbedingungen nur in einer Skriptsprache möglich sind. Diese Anpassungen werden in der jeweils erforderlichen Sprache durchgeführt.

5 Formulierung des Konzeptes für ein integriertes Produkt- und Projektmanagement

In den vorherigen Kapiteln wurden Architektur und Arbeitsweise von Projektmanagement- und Produktdatenmanagement-Systemen diskutiert und die wesentlichen Defizite aufgezeigt, die diese beiden EDV-Systeme bei der Unterstützung von Produktentwicklungsprozessen zeigen.
Im Folgenden soll nun unter Berücksichtigung der in Kapitel 3 formulierten Anforderungen ein Konzept für eine integrierte Lösung entwickelt werden. Besonderes Augenmerk wird dabei auf die Entwicklung einer Benutzeroberfläche gelegt, die alle wesentlichen Informationen auf einen Blick aufbereitet. Da diese virtuelle Plantafel den Brückenschlag zwischen PDM- und PM-Systemen realisiert, wird sie im Folgenden als P2-Plantafel bezeichnet.

5.1 Lösungsansatz

Wie bereits in Kapitel 3.1.2 erörtert wurde, besitzen Fertigungsunternehmen in der Regel eine an das Unternehmen angepasste PDM-Lösung. Ein Systemwechsel zu einer neuentwickelten Plantafel wäre aufgrund des damit verbundenen Anpassungs- und Einarbeitungsaufwands für deren PDM-Komponente nicht wirtschaftlich.
Weiterhin kann die Möglichkeit der Eigenentwicklung eines Systems mit PDM-Funktionalität aufgrund der Komplexität dieser Thematik und dem damit verbundenen Programmieraufwand, nicht in Betracht gezogen werden.
Aus diesen Gründen besteht der einzige praxistaugliche Lösungsansatz zur Optimierung der EDV-Unterstützung von Produktentwicklungsprojekten in der Integration des Projektmanagements in vorhandene PDM-Systeme.

Die folgenden Abbildungen verdeutlichen den Nutzen dieser integrierten EDV-Unterstützung gegenüber einer herkömmlichen EDV-Unterstützung. Bei der herkömmlichen EDV-Unterstützung werden Projektmanagement und Projektabwicklung innerhalb getrennter Systeme abgewickelt zwischen denen im Idealfall ein manueller Datenabgleich durchgeführt wird (Abbildung 5-1). Diese Konstellation birgt naturgemäß das Risiko eines trägen, inkonsistenten Informationbestandes. Währenddessen vereint das Konzept der P2-Plantafel beide Teilsysteme zu einer durchgängigen Lösung (vgl. Abbildung 5-2).

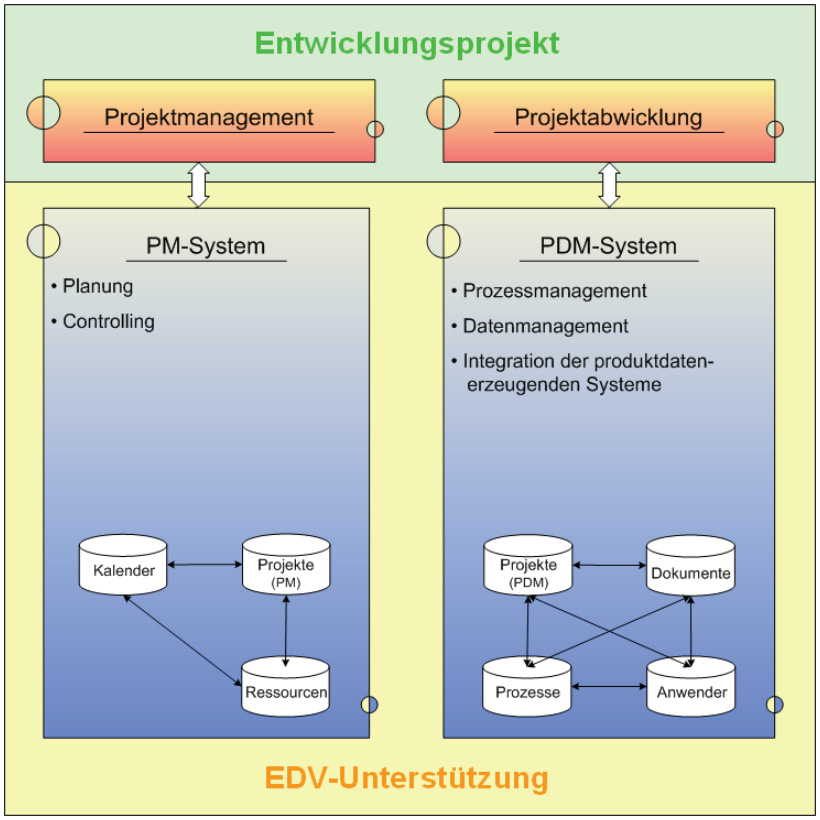


Abbildung 5-1: Herkömmliche EDV-Unterstützung

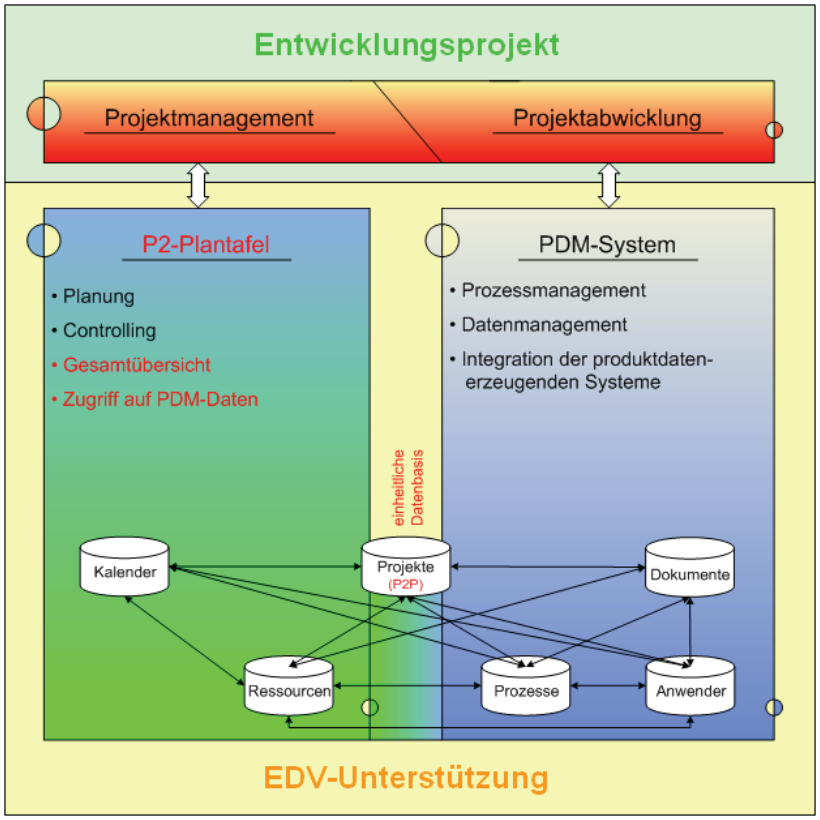


Abbildung 5-2: Integrierte EDV-Unterstützung

Bei dem neu entwickelten Konzept werden im Gegensatz zu herkömmlichen Planungswerkzeugen bei der Planung und Steuerung von Entwicklungsprojekten alle relevanten Informationen aus dem PDM-System berücksichtigt. Weiterhin können sämtliche Planungsergebnisse direkt in das PDM-System übertragen werden. Zudem unterstützt das integrierte System ein umfassendes Controlling in Echtzeit.
Diese Aspekte kennzeichnen eine entscheidende Verbesserung im Vergleich zu der derzeitig verfügbaren EDV-Unterstützung.

Die Integration von Produktdatenmanagement und Projektmanagement zu einer durchgängigen EDV-Lösung steht im Zentrum des Plantafel-Konzeptes. Weiterhin sind bei der Konzeption Benutzerfreundlichkeit und Anpassungsfähigkeit als zentrale Randbedingungen zu berücksichtigen (vgl. Anforderungen, Kap.3).
Der Benutzerfreundlichkeit wird durch die Erstellung einer übersichtlichen und intuitiv bedienbaren Arbeitsoberfläche Rechnung getragen, welche sämtliche relevanten Informationen lückenlos aufbereitet. Die komplette Produktentwicklung kann dadurch mittels einer einzigen Benutzeroberfläche geplant und gesteuert werden. Die Anpassungsfähigkeit der Plantafel wird durch Verwendung einer geeigneten Anwendungsarchitektur realisiert.
In den nachfolgenden Kapiteln wird zunächst ein Vergleich von PM- und PDM-Systemen auf funktionaler Ebene durchgeführt, bevor die Ausarbeitung einer konkreten Anwendungsarchitektur zur Integration dieser beiden Systeme erfolgt.

5.2 Funktionale Analyse

Da bei der P2-Plantafel eine Vernetzung von zwei unterschiedlichen Typen von EDV-Systemen anstrebt wird, ist vor der Konzeptionierung einer konkreten Anwendungsarchitektur eine umfassende Analyse der beiden zu integrierenden Systeme erforderlich.
Die Abbildungen 5-3 und 5-4 zeigen die zentralen Objekte der beiden unterschiedlichen Systemtypen. Bei der Auswertung des Stands der Technik wurden die Objekte und Abhängigkeiten der Einzelsysteme bereits hinreichend thematisiert (vgl. Kapitel 2). An dieser Stelle erfolgt nun der direkte Vergleich. Zur Verdeutlichung von Gemeinsamkeiten sind gleichartige Objekte in den Abbildungen in derselben Farbe hervorgehoben.

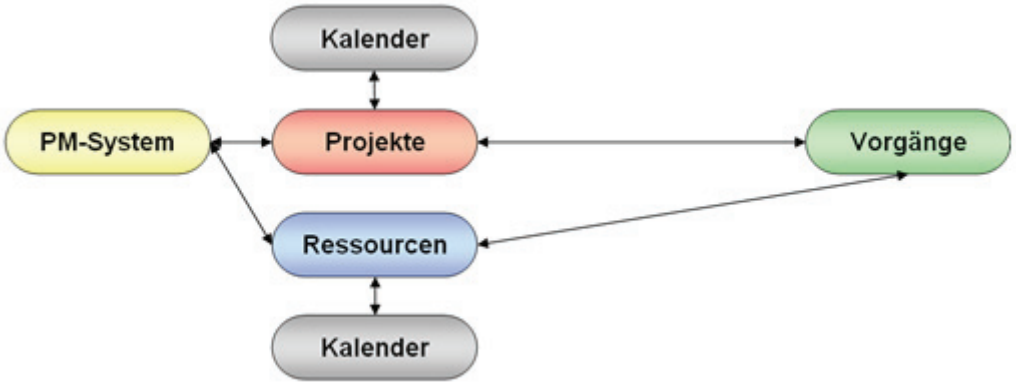


Abbildung 5-3: zentrale Objekte eines PDM-Systems

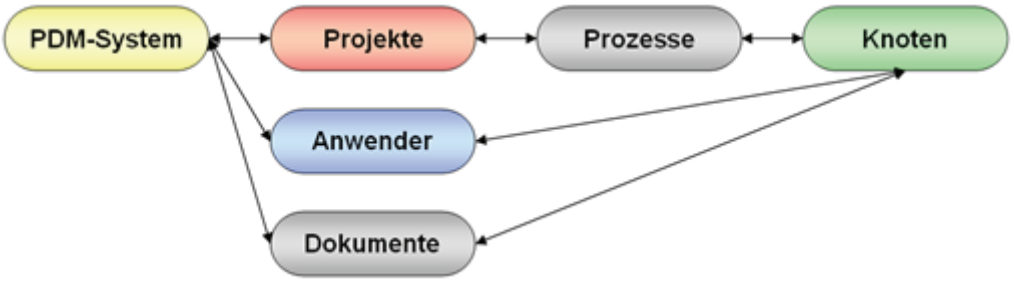


Abbildung 5-4: zentrale Objekte eines PM-Systems

Das ***Projekt*** stellt sowohl bei PDM- als auch bei PM-Systemen das zentrale Objekt dar. Mit PDM-Systemen werden Entwicklungs- oder Änderungsprojekte verwaltet, während mit Hilfe von PM-Systemen beliebige Projekte abgewickelt werden können. Weiterhin erfolgt in beiden Systemen eine Unterteilung der durchzuführenden Arbeitsabläufe in einzelne Arbeitsschritte. Diese werden im PDM-Umfeld als ***Knoten*** bezeichnet, während sie beim Projektmanagement ***Vorgänge*** heißen. Eine weitere Gemeinsamkeit besteht darin, dass in beiden Systemen Kapazitäten erfasst werden, die zur Abwicklung der einzelnen Arbeitsschritte erforderlich sind. Während in PDM-Systemen in der Regel ausschließlich ***Anwender*** abgebildet werden, erfolgt im PM-Umfeld eine Zusammenfassung von Mitarbeitern und Maschinen zu allgemeingültigen ***Ressourcen***.

Das Vorhandensein dieser gleichartigen Objekte bildet die Grundlage für die Vernetzung von PDM- und PM-Systemen. Die übrigen, nur bei einem Systemtyp auftretenden Objekte, sind durch die unterschiedlichen Aufgabenbereiche der beiden Systemtypen bedingt. Da PDM-Systeme der Verwaltung aller im Laufe von Produktentwicklungen anfallenden Daten dienen, stehen ***Dokumente*** neben Arbeitsabläufen im Fokus des PDM-Systems. PM-Systeme dienen hingegen hauptsächlich der Planung und dem Controlling von Projekten. Ihre Funktionalität erfordert daher keine Dokumente. Stattdessen werden von PM-Systemen verschiedene ***Kalender*** benötigt, welche die Grundlage für eine Zeitplanung bilden.

Ein weiterer Unterschied zwischen den beiden Anwendungssystemen besteht in der Verwaltung der Arbeitsabläufe. PM-Systeme administrieren sämtliche Arbeitsabläufe auf Projektebene. Die einzelnen Arbeitsschritte des Projektes (***Vorgänge***) werden durch Angabe von Vorgangsbeziehungen und einer Gliederungsstufe den unterschiedlichen Arbeitsabläufen zugewiesen.
Währenddessen verwenden PDM-Systeme ein dem Projekt untergeordnetes Objekt zur Verwaltung der einzelnen Arbeitsabläufe. Dieses wird als ***Prozess*** bezeichnet. Ein PDM-Projekt kann mittels von Prozessen eine beliebige Anzahl von Arbeitsabläufen verwalten, die sich aus beliebig vielen Arbeitsschritten (***Knoten***) zusammensetzen.

Der initiale Schritt bei der Konzeption der P2-Plantafel besteht darin, die zentralen Objekte von PM- und PDM-Systemen zu fusionieren. In Abbildung 5-5 ist der grundlegende Aufbau einer integrierten Struktur dargestellt.

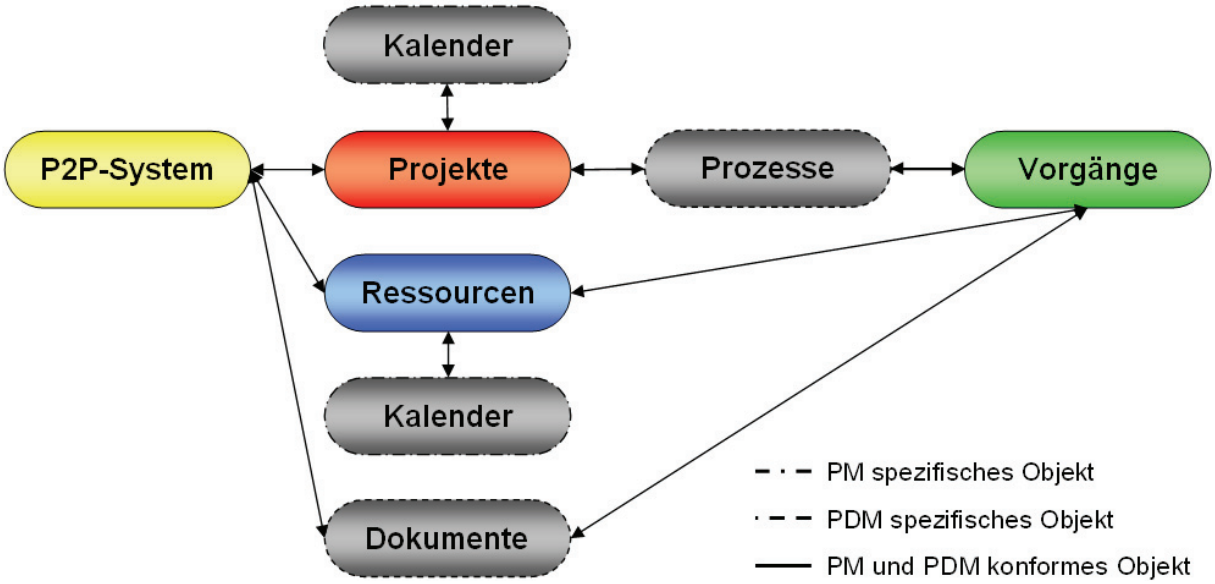


Abbildung 5-5: zentrale Objekte der P2-Plantafel

Das Kernstück für die Kombination von Projektmanagement- und Produktdatenmanagement-Systemen sind die zuvor diskutierten gleichartigen Objekte. Da in beiden Systemen für vergleichbare Objekte unterschiedliche Bezeichnungen etabliert sind, wird an dieser Stelle festgelegt, dass für die P2-Plantafel die Termina aus dem Projektmanagement verwendet werden.
Neben den gleichartigen Objekten ***Projekt***, ***Vorgang*** und ***Ressource***, müssen die PM-spezifischen ***Kalender*** sowie die PDM-spezifischen ***Prozesse*** und ***Dokumente*** in dem Konzept der P2-Plantafel berücksichtigt werden.

Nachfolgend wird eine Anwendungsarchitektur zur Realisierung der P2-Plantafel vorgestellt, welche unter Berücksichtigung der vorliegenden Randbedingungen eine Umsetzung dieser Struktur ermöglicht.

5.3 Integrationsstrategie

Die technische Realisierung eines integrierten Produktdaten- und Projektmanagement lässt sich mittels zwei verschiedener Lösungsstrategien verwirklichen:

Die ***erste*** Lösungsvariante besteht darin, die P2-Plantafel in Form eines Moduls direkt in ein PDM-System zu integrieren (vgl. Abbildung 5-6, links). Diese Lösung zeichnet sich durch eine hohe Integrationstiefe aus. Diese Architektur ist allerdings durch eine geringe Flexibilität gekennzeichnet. Die Implementierung der Plantafel würde nur für ein spezifisches PDM-System Gültigkeit besitzen. Aus diesem Grund ist der erste Lösungsansatz für ein allgemeingültiges Konzept nicht geeignet.

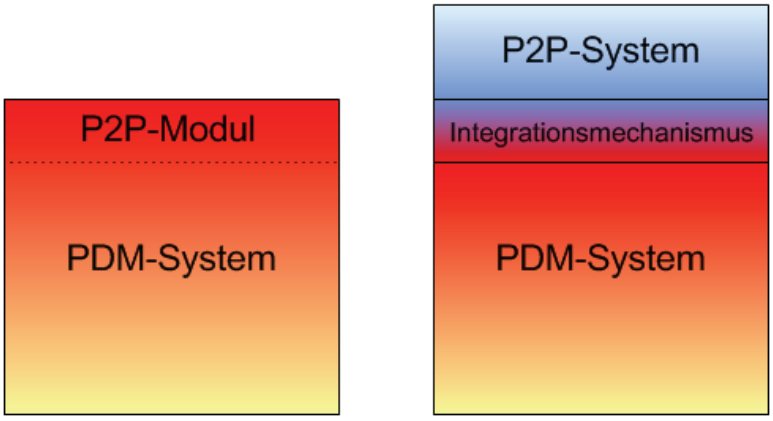


Abbildung 5-6: Integrationsstrategien

Der ***zweite*** Lösungsansatz besteht darin, ein externes Plantafel-System mittels geeigneter Integrationsmechanismen in ein PDM-System zu integrieren (vgl. Abbildung 5-6, rechts). Diese Lösungsstrategie bietet gegenüber Lösungsvariante 1 den Vorteil, dass die Plantafel theoretisch mit jedem PDM-System kombiniert werden kann. Dazu ist in jedem verwendeten PDM-System lediglich die Implementierung von systemspezifischen Integrationsmechanismen durchzuführen. Somit wird die für ein allgemeingültiges Konzept erforderliche Anpassungsfähigkeit realisiert. Aus diesen Gründen wird diese Lösungsstrategie für die Konzeptionierung des integrierten Produktdaten- und Projektmanagements verwendet.
Vor der technischen Realisierung des ausgewählten Lösungsansatzes ist weiterhin zu entscheiden, ob die P2-Plantafel als funktionale Erweiterung eines bestehenden Projektmanagementsystems implementiert wird oder ob eine Neuentwicklung vorteilhaft ist.

Wie in der funktionalen Analyse beschrieben wurde, sind die Datenstrukturen von PM- und PDM-Systemen aufgrund der unterschiedlichen Einsatzbereiche grundverschieden. Demzufolge sind für die Anpassung eines PM-Systems gravierende Änderungen in dem Datenmodell und den Berechnungsalgorithmen erforderlich.
Weiterhin besteht ein wesentlicher Aspekt der P2-Plantafel in der Bereitstellung einer Arbeitsoberfläche, die sämtliche zentralen PDM- und PM-Daten übersichtlich aufbereitet und Zugriff auf integrationsspezifische Funktionen liefert. Für die Realisierung der Plantafel müssen demzufolge große Teile der Benutzeroberfläche des PM-Systems neu gestaltet werden.

Aus den dargelegten Gründen sind PM-Systeme als Komponenten eines Plantafel-Systems ungeeignet. Die Realisierung dieses Lösungsansatzes würde derart gravierende Änderungen in dem PM-System erfordern, dass der daraus resultierende Aufwand den zu erwartenden Nutzen bei weitem übersteigen würde. Daher erfolgt im Rahmen dieser Dissertation die Neuentwicklung eines Systems mit PM-Funktionalität.
Da die Programmlogik von PM-System im Vergleich zu PDM-Systemen eine wesentlich geringere Komplexität aufweist, ist die Realisierbarkeit dieses Lösungsansatzes gewährleistet.

In den nachfolgenden Kapiteln erfolgt die Ausarbeitung eines konkreten Konzeptes, das auf der dargelegten Integrationsstrategie beruht.

5.4 Systemarchitektur

Grundlage für das Konzept für ein integriertes Produktdaten- und Projektmanagement ist ein Drei-Schichten-Modell. Diese Systemarchitektur zeichnet sich dadurch aus, dass die grundlegenden Programmfunktionen logisch voneinander getrennt sind. Somit können die einzelnen Schichten unabhängig voneinander implementiert und gewartet werden. Dadurch wird eine flexible Programmentwicklung realisiert. Zudem ermöglicht die Kapselung in funktionale Schichten die Konzeptionierung eines anpassungsfähigen Systems, das abhängig vom konkreten Anwendungsfall skaliert werden kann.
Weiterhin unterstützt die verwendete Systemarchitektur die Integration der Plantafel mit unterschiedlichen PDM-Systemen sowie die Nutzung der Projektmanagementfunktionalitäten ohne Anbindung an ein PDM-System. Diese Betriebsmodi werden im Folgenden als Integrationsmodus und Stand-Alone-Modus bezeichnet.

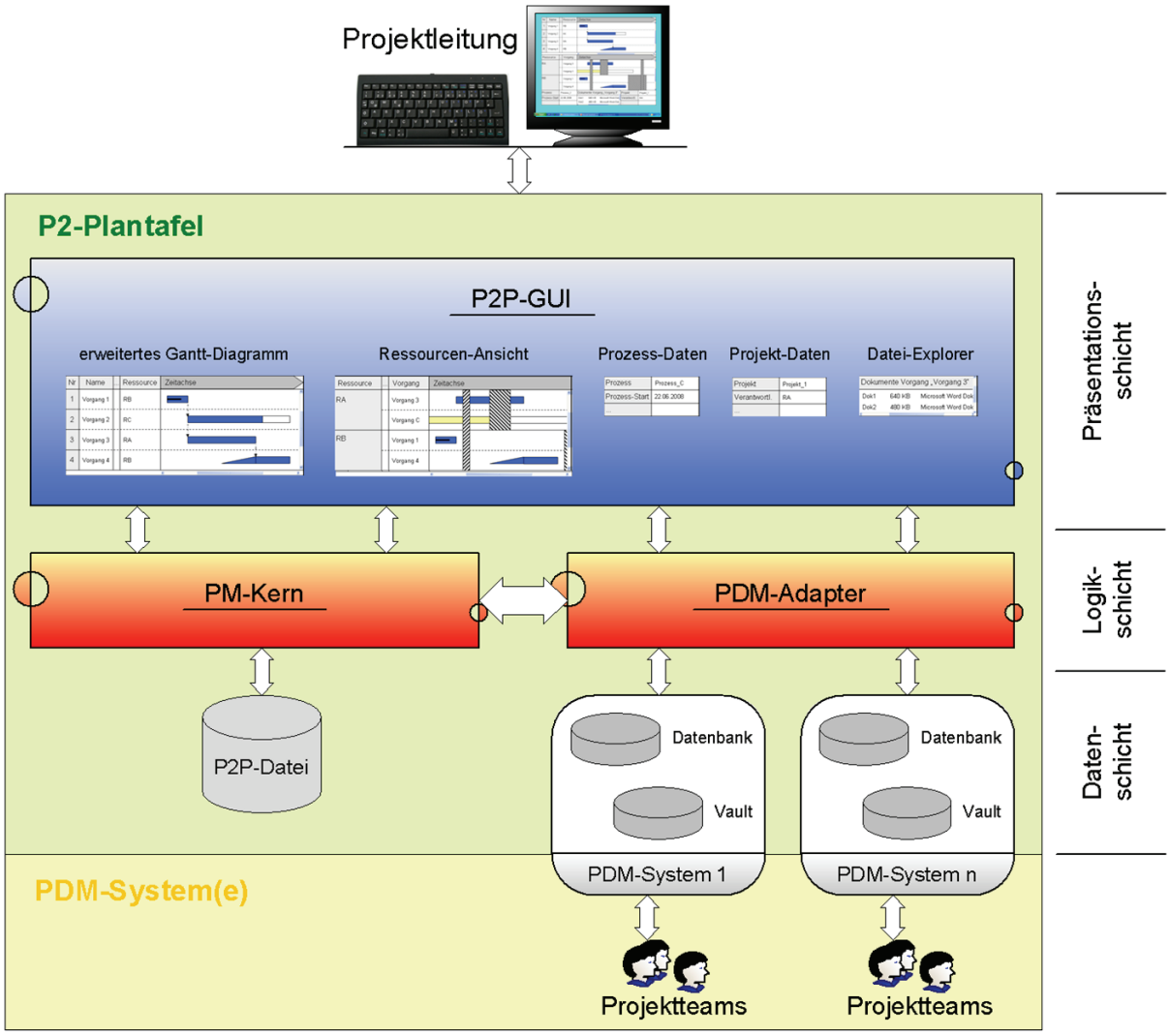


Abbildung 5-7: Systemarchitektur der P2-Plantafel

Die einzelnen Schichten der Systemarchitektur werden nachfolgend erläutert.

Präsentationsschicht

Die Präsentationsschicht fungiert als Benutzerschnittstelle. Ihre Funktionalität umfasst im Wesentlichen die Repräsentation der Daten und die Benutzerinteraktion. Um der Anforderung nach einer übersichtlichen grafischen Benutzeroberfläche (GUI) gerecht zu werden, setzt sich die Plantafel aus fünf verschiedenen Ansichten zusammen. Dies ermöglicht eine übersichtliche Darstellung sämtlicher relevanten Daten zu dem aktiven Projekt (***Projekt-Daten***), dem aktuellen Prozess (***Prozess-Daten***) und den dazugehörigen Vorgängen (***Gantt-Diagramm***). Weiterhin können in demselben Arbeitsbereich die verfügbaren Ressourcen (***Ressourcen-Ansicht***) und die verknüpften PDM-Dateien (***Datei-Explorer***) angezeigt werden. Ferner sind zur Verarbeitung von Benutzereingaben entsprechende Steuerelemente, Dialoge und Menüs verfügbar.

Logikschicht

Die Logikschicht konzentriert die Programmlogik in Form von Verarbeitungsroutinen und koordiniert den Zugriff auf sämtliche Daten. Zu diesem Zweck ist sie in zwei Module untergliedert: PM-Kern und PDM-Adapter.

Die Aufgabe des **PM-Kerns** besteht darin, eine Klassenhierarchie für das Projektmanagement bereitzustellen, die Attribute zur Erfassung sämtlicher relevanten Informationen beinhaltet und Methoden zur Verarbeitung dieser Attribute implementiert. Die Anforderung, dass die PM-Komponente der Plantafel auch ohne PDM-Integration nutzbar sein soll, wird berücksichtigt indem der PM-Kern über Methoden zur Serialisierung verfügt. Planungsergebnisse können dadurch vom PDM-System unabhängig in einer eigenständigen Datei gespeichert werden. Dies ermöglicht es die Systemeinführung der P2-Plantafel zweistufig durchzuführen. Zunächst kann die P2P als klassische PM-Anwendung verwendet werden (Stand-Alone-Modus), bevor sie durch Kopplung zu einem PDM-System zu einer integrierten PDM-PM-Anwendung erweitert wird (Integrationsmodus).

Der **PDM-Adapter** implementiert die Schnittstelle zwischen dem Stand-Alone-Modus und den Daten des PDM-Systems. Er bildet somit die Schlüsselkomponente bei der Integration von Produktdatenmanagement und Projektmanagement. Zu diesem Zweck ist der PDM-Adapter in zwei Unterschichten unterteilt. Die obere Schicht definiert PDM-neutrale Routinen zum Datenaustausch, während die untere Adapterschicht API-spezifische Merkmale des verwendeten PDM-Systems bzw. der verwendeten PDM-Systeme berücksichtigt. Aufgrund dieser Unterteilung ist zur Anpassung der P2-Plantafel an ein spezifisches PDM-System lediglich eine Anpassung der unteren Adapterschicht erforderlich.

Datenschicht

Die Datenschicht der Anwendung wird durch Dateisysteme und Datenbanken gebildet. Im Stand-Alone-Modus werden die anfallenden P2P-Dateien direkt im Dateisystem abgelegt. Befindet sich die Anwendung im Integrationsmodus sind P2P-Dateien nicht erforderlich, weil die Datenverwaltung über das zuständige PDM-System erfolgt. In der Datenbank des PDM-Systems werden die Metadaten[27] der PDM-Daten gespeichert, während die zugehörigen Dateien in einem vom PDM-System verwalteten Bereich des Dateisystems abgelegt werden (Vault). Auf wie viele Vaults und Datenbanken ein spezifisches PDM-System zurückgreift ist für die Integration unerheblich, weil der Zugriff auf die im PDM-System gespeicherten Daten ausschließlich über die API des betreffenden PDM-Systems erfolgt.

[27] Metadaten: beschreibende Zusatzdaten einer Datei (z.B. Identifikationsnummer, Revision, Status)

In den folgenden Unterkapiteln wird die Entwicklung des P2P-Konzeptes vollzogen. Um vor der Integration der P2-Plantafel mit einem konkreten PDM-System die fehlerfreie Funktion des PM-Modus zu gewährleisten, wird der Entwicklungsvorgang in zwei Stufen unterteilt: Zunächst erfolgt die Entwicklung der Komponenten für den Betrieb im Stand-Alone-Modus (***Stufe 1***). Im Anschluss daran wird die PDM-Integration ausgearbeitet (***Stufe 2***).

Abbildung 5-8 stellt die beiden Entwicklungsstufen grafisch dar. Bei der Konzeptionierung der beiden Stufen wird konsequenterweise von Innen nach Außen vorgegangen. Ausgehend von der Formulierung der Datensätze (Datenschicht) werden die Zugriffs- und Manipulationsmöglichkeiten entwickelt (Logikschicht), woraufhin schließlich die Ausarbeitung der Benutzerschnittstelle erfolgt (Präsentationsschicht).

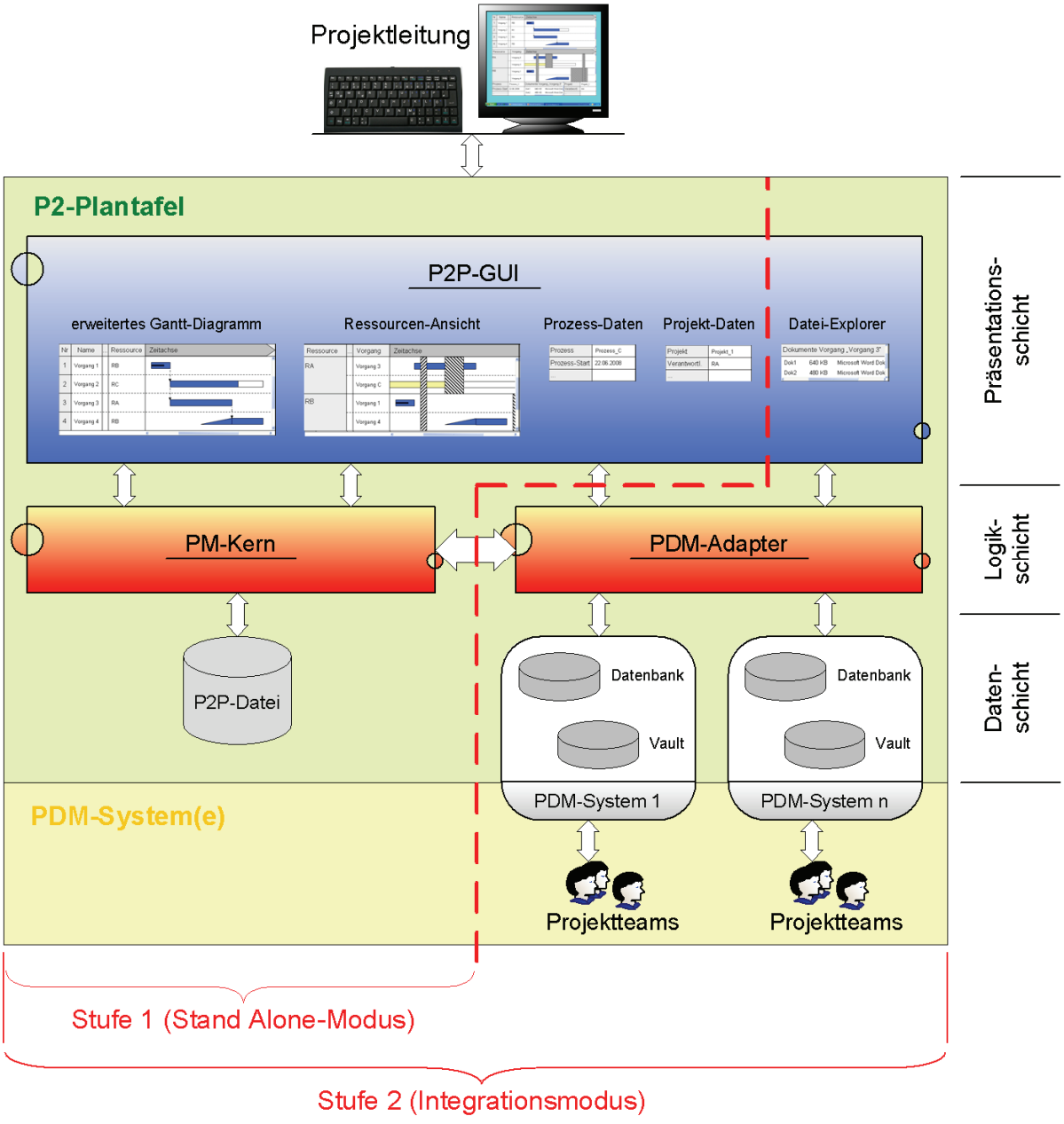


Abbildung 5-8: Entwicklungsstufen der P2P

5.5 Stand-Alone-Modus

Im Zuge von Kapitel 4 wurden bereits die Randbedingung und die Softwaretechnologie, die bei der Konzeption der P2P zu berücksichtigen sind, einer eingehenden Betrachtung unterzogen. Die grundlegende Architektur der Plantafel ist dadurch determiniert, dass die Programmentwicklung auf Grundlage der objektorientierten, betriebssystemnahen Programmierung (OOB) erfolgt und als Plattform das Windows-Betriebssystem verwendet wird.
Für dieses Betriebssystem hat sich eine Softwarearchitektur durchgesetzt, die von der überwiegenden Mehrheit der gängigen OOB-Anwendungen eingehalten wird. Dabei handelt es sich um die auf dem MFC-Framework beruhende Dokumenten-Ansicht-Architektur.

Abbildung 5-9 verdeutlich die Struktur einer Anwendung, welche auf der Dokumenten-Ansicht-Architektur beruht. Die Datenverarbeitung ist dort in zwei Klassen gekapselt: Dokumenten- und Ansichtklasse. Die zu bearbeitenden Daten werden von einem Objekt der Dokumentklasse repräsentiert, während die grafische Darstellung dieser Daten sowie die Benutzerinteraktionen von mindestens einem Ansichtsobjekt abgewickelt wird. Durch diese Architektur wird eine Trennung von der Datenrepräsentation und der Datenadministration erzielt. Dies ermöglicht es, die Daten des Dokumentes in unterschiedlichen Ansichten anzuzeigen und somit von denselben Daten verschiedene Sichtweisen darzustellen. Als Beispiel sei ein Textdokument in MS Word genannt. Für denselben Fließtext sind sowohl eine Seitenlayout-Ansicht als auch eine Gliederungs- und eine Weblayoutansicht verfügbar. Der Anwender kann zur Laufzeit zwischen den einzelnen Ansichten wechseln. Dies ist möglich, da die Ansichtsobjekte lediglich für die grafische Darstellung der Dokumentdaten verantwortlich sind und sämtliche Änderungen an den Daten in dem Dokumentobjekt vorgenommen werden. Ein übergeordnetes Steuerprogramm ist dafür verantwortlich, dass bei geänderten Dokumentdaten die betroffenen Ansichten aktualisiert werden.

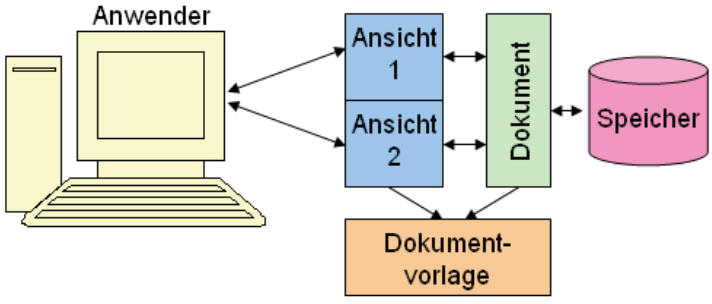


Abbildung 5-9: Dokumenten-Ansicht-Architektur

Der komplexe Vorgang des Erstellens eines Dokumentes, mit den zugehörigen Ansichten und Rahmenfenstern, wird von der Dokumentvorlage abgewickelt.

Anwendungen, die unterschiedliche Dokumenttypen unterstützen, müssen für jeden Typ eine eigene Dokumentvorlage bereitstellen. Als Beispiel sei an dieser Stelle die CAD-Anwendung SolidWorks angegeben, die sowohl die Erstellung von Einzelteil- (.sldprt) als auch von Baugruppendokumenten (.sldasm) unterstützt.

Zur Erstellung von Dokumentvorlagen stellt die MFC zwei unterschiedliche Basisklassen bereit. Dies ist darauf zurückzuführen, dass Windows-Anwendungen in SDI[28]- und MDI[29]-Anwendungen unterteilt werden. Während SDI-Applikationen lediglich ein Dokument pro Programmfenster bereitstellen, unterstützen MDI-Anwendungen das zeitgleiche Öffnen von mehreren Dokumenten in einem Programmfenster.
Abbildung 5-10 verdeutlicht diesen Sachverhalt anhand des direkten Vergleiches zwischen einer SDI-Anwendung (Windows-Explorer, links) und einer MDI-Anwendung (SolidWorks, rechts). Beide Anwendungen stellen die Daten ihrer Dokumente in einer zweigeteilten Arbeitsoberfläche dar, die jeweils aus einer Baumstruktur-Ansicht und einer weiteren Ansicht bestehen. Der wesentliche Unterschied zwischen diesen beiden Anwendungen besteht darin, dass bei der SDI-Anwendung beim Öffnen jedes Dokumentes ein neues Programmfenster erstellt wird, während bei der MDI-Anwendung ein neues Rahmenfenster in das bestehende Programmfenster eingefügt wird. Dabei kann zu jeder Zeit immer nur ein Rahmenfenster bzw. Dokument aktiv sein.

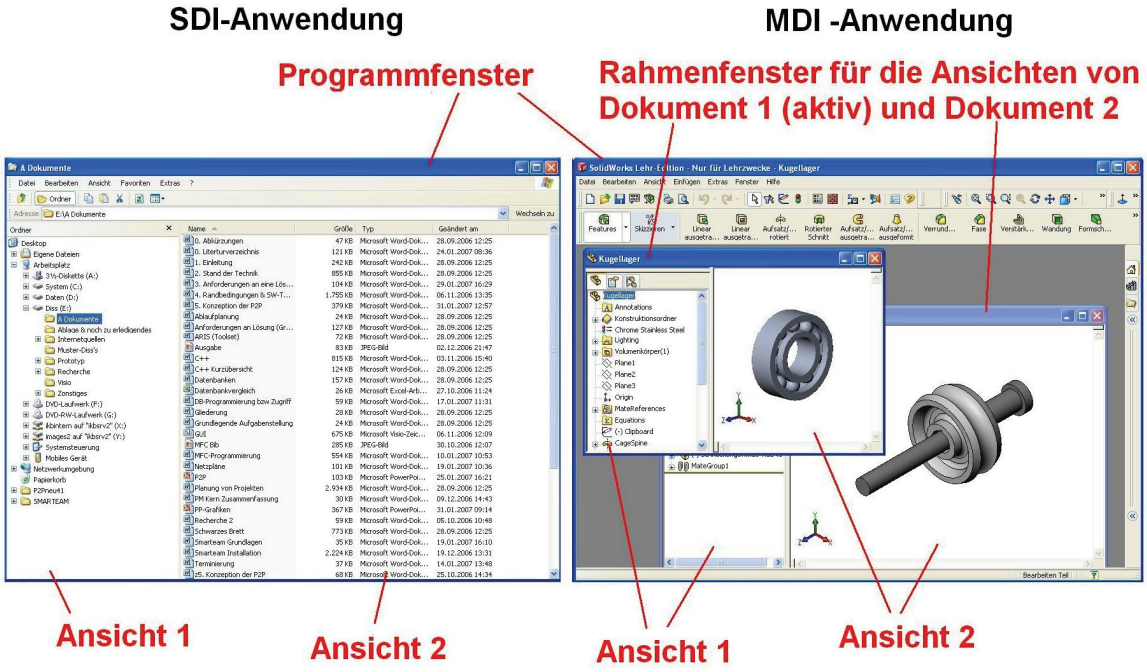


Abbildung 5-10: SDI vs. MDI

[28] SDI: Single Document Interface

[29] MDI: Multiple Document Interface

Die Dokumenten-Ansicht-Architektur ist für die Konzeptionierung des Stand-Alone-Modus ein logischer Ansatzpunkt. Sie stellt die für den objektorientierten Ansatz des P2P-Konzeptes notwendigen, allgemeingültige Basisklassen bereit und bietet weiterhin die Möglichkeit geeignete Spezialisierungen von ihnen abzuleiten.

Vor der Konkretisierung der P2P-Anwendungsaritektur ist zunächst die Festlegung der elementaren Bestandteile des Plantafel-Konzeptes und ihrer charakteristischen Eigenschaften erforderlich. Dabei werden zunächst die Elemente, die für das Projektmanagement im Stand-Alone-Modus erforderlich sind betrachtet.

Im Mittelpunkt des Projektmanagements stehen drei zentrale Elemente:
• das Projekt,
• der Vorgang
• und die Ressource.

Weiterhin werden als ergänzende Informationen
• die Kapazität bzw. das Leistungsvermögen der Ressourcen sowie
• die Zeit in Form eines Kalenders benötigt.

Außerdem sind bei Konzeption des Stand-Alone-Modus Prozesse zu berücksichtigen, weil diese für die nachfolgende PDM-Integration erforderlich sind.

Die Erfassung von Kosteninformationen wird an dieser Stelle bewusst außer Acht gelassen, da die P2-Plantafel als Werkzeug zur Unterstützung der Produktentwicklung und nicht als Hilfsmittel für die Finanzbuchhaltung konzeptioniert wird.
Die erarbeitete Anwendungsarchitektur unterstützt jedoch eine nachträgliche Erweiterung des Plantafel-Konzepts, falls zu einem späteren Zeitpunkt die Integration einer Kostenkalkulation angestrebt werden sollte. Dazu sind die Klassen der P2-Plantafel um betriebswirtschaftliche Attribute, wie zum Beispiel den Maschinenstundensatz, zu ergänzen. Weiterhin wären im Zuge dieser Erweiterung der P2-Plantafel eine Ausdehnung der Berechnungsalgorithmen sowie eine Anpassung der Benutzeroberfläche erforderlich.

Es folgt eine kurze Beschreibung der grundlegenden Elemente des Stand-Alone-Modus der P2-Plantafel.

Projekt

Wie im Rahmen von Kapitel 2 bereits ausgeführt wurde, wird im Rahmen dieser Arbeit ein Projekt als einmaliges Vorhaben verstanden, welches eine gewisse Komplexität aufweist und bei dem besondere Anforderungen an die zeitliche Abwicklung sowie an die zu erzielende Qualität vorliegen.

Abgesehen von wenigen globalen Attributen sind alle Elemente der P2-Plantafel einem oder mehreren Projekten untergeordnet. Das Projekt bildet aus diesem Grund die oberste Hierarchieebene eines jeden Planungsvorhabens.

Im Sinne der Optimierung der EDV-Unterstützung von Produktentwicklungsprojekten berücksichtig die Architektur der P2-Plantafel, dass PDM-Projekte in der Regel mehrere Arbeitsabläufen (Prozesse) beinhalten.

Prozess

Ein Prozess ist ein Arbeitsablauf, der dadurch gekennzeichnet ist, dass er sich aus einer Reihe von Teilaufgaben (Vorgängen) zusammensetzt, die in Summe zur Erzielung eines definierten Ergebnisses führen.

Vorgang

Ein Vorgang ist ein Vorhaben, das Zeit erfordert und dessen Ziel im Erreichen eines definierten Endergebnisses besteht. Er wird durch einen definierten Anfang und ein definiertes Ende begrenzt. Die Vorgangsdauer resultiert aus der Differenz zwischen diesen beiden Ereignissen.

Vorgänge sind die zentralen Elemente des Projektmanagements. Die Festlegung der zeitlichen Lage der einzelnen Vorgänge aller Prozesse sowie die Berechnung der Vorgangsdauern stehen im Mittelpunkt des Projektmanagements. Die Berechnungsalgorithmen sowie die zu berücksichtigen Randbedingungen werden zu einem späteren Zeitpunkt konkretisiert.

Ressource

Ressourcen[30] sind Kapazitäten, die zur Bearbeitung von Vorgängen erforderlich sind. Es wird zum Beispiel das Vorgangsziel „Konstruktion einer Abdeckplatte“ erreicht indem die Ressource „Konstrukteur1“ seine Arbeitskraft in diesen Vorgang investiert. Die Planung von Ressourcen ist notwendig, wenn eine eingesetzte Ressource knapp oder kostenintensiv ist.

Da Projekte in der Praxis immer über eine endliche Anzahl von Mitarbeitern verfügen, ist die zentrale Ressource der Mensch bzw. der Mitarbeiter. Sollten die dem Projekt zur

[30] Ressource: <franz.> Hilfsmittel [12]

Verfügung stehenden Maschinen bzw. Gerätschaften ebenfalls knapp oder kostenintensiv sein, sind sie in der Projektplanung zu berücksichtigen. Dabei kann der Maschinenbediener ggf. als „Zubehör“ zur Maschine angesehen werden, so dass lediglich eine Planung der maschinellen Ressourcen erforderlich ist.

Kapazität

Zur Verbesserung von Planungsgenauigkeit und Termintreue beinhaltet die P2-Plantafel ein neuartiges Konzept für die Erfassung von Ressourcenqualifikation und –leistungsfähigkeit.

Jede Ressource verfügt über spezifische für die Planung relevante Eigenschaften. Diese werden im folgenden Kapazität genannt. Die Kapazität einer Ressource bestimmt, ob sie die Eigenschaften besitzt, die für die Bearbeitung eines Vorgangs erforderlich sind. Außerdem übt die Kapazität direkten Einfluss auf die zur Abwicklung eines Vorgangs benötigte Zeit aus.

Kalender / Leerlauf

Zur Planung der zeitlichen Lage der Vorgänge eines Projektes sind Kalender erforderlich. Für jedes Projekt gibt es einen globalen Kalender (den sogenannten Projektkalender) welcher der Erfassung der Werktage des Projektes dient. Weiterhin benötigt jede Ressource einen lokalen Kalender, in der die Arbeitstage der Ressource sowie die Anzahl der Arbeitsstunden berücksichtigt werden (Ressourcenkalender). Durch Kombination dieser beiden Kalender kann genau erfasst werden, wann und mit wie vielen Arbeitsstunden jede Ressource dem Projekt zur Verfügung steht.

Bei humanen Ressourcen sind arbeitsfreie Zeiten dadurch bedingt, dass ein Mitarbeiter nur eine begrenzte Anzahl an Stunden pro Tag arbeitet, aber auch durch Wochenenden, Feiertage, Urlaub, Krankheit und ähnliche Ausfalltage. Maschinelle Ressourcen können hingegen im Idealfall 24 Stunden am Tag arbeiten, haben aber bedingt durch Wartungsintervalle und Schäden ebenfalls Ausfallzeiten. Um für die Zeitintervalle in denen die Leistung einer Ressource nicht für die Bearbeitung von Vorgängen zur Verfügung steht einen einheitlichen Terminus zu gebrauchen, wird dafür im Folgenden der Begriff Leerlaufzeit bzw. Leerlauf verwendet.

Nach der Vorstellung der elementaren Bestandteile des Plantafel-Konzeptes erfolgt nun die Detaillierung des Konzeptes für den Stand-Alone-Modus. Wie einleitend bereits dargelegt wurde, wird dabei von Innen nach Außen vorgegangen.

5.5.1 Datenschicht

Im Stand-Alone-Modus werden zur dauerhaften Speicherung von Projektdaten Anwendungsdateien verwendet. Um diese im Dateisystem eindeutig zu identifizieren, erhalten diese die Dateiendung .p2p. Auf diese Weise können die Anwendungsdateien der Plantafel leicht von den Dateien anderer Anwendungen unterschieden werden.
Das Speichern und Laden der Anwendungsdateien beruht auf dem Prinzip der Serialisierung. Diese Funktion der objektorientierten Programmierung ermöglicht es, den aktuellen Zustand eines Objektes in einen Datenstrom (Stream) zu konvertieren und somit in eine speicherungsfähige Form zu bringen. Der Stream kann danach dauerhaft gespeichert werden indem er mit Hilfe entsprechender Archivierungsfunktionen in eine Datei geschrieben oder in eine Datenbank abgelegt wird. Durch die Umkehrung dieses Prozesses können gespeicherte Zustände von Objekten jederzeit wiederhergestellt werden. Dieser Vorgang wird als Deserialisierung bezeichnet[31].
Da das Speichern und Laden der Daten der P2-Plantafel durch Aufruf der Serialisierungsfunktion der im PM-Kern implementierten Klassen erfolgt, ist im Stand-Alone-Modus keine explizite Definition einer Datenstruktur erforderlich.

5.5.2 Logikschicht

Die Anwendung des Projektmanagements zur Planung und Abwicklung von Projekten erfordert es, Projekte und deren Eigenschaften zu erfassen. Die Ausarbeitung der Elemente, die für die vollständige Erfassung sämtlicher relevanten Informationen erforderlich sind sowie die Charakterisierung der vielfältigen Beziehungen zwischen den unterschiedliche Elementen kann nur nach den Prinzipien der objektorientierten Programmierung erfolgen. Zur umfassenden Darstellung der Elemente des entwickelten Konzeptes sowie deren Beziehungen werden nachfolgend zwei verschiedene Diagrammtypen verwendet: das Klassen- und das Objektdiagramm. Das Klassendiagramm dient dazu die statische Struktur des Anwendungssystems wiederzugeben. Dazu beschreibt es den Aufbau und die Beziehungen der vorhandenen Klassen. Im Gegensatz dazu dient das Objektdiagramm der Abbildung des dynamischen Verhaltens des Anwendungssystems. Zu diesem Zweck gibt es die Objekte, die zu einem bestimmten Zeitpunkt vorhanden sind, und deren Beziehungen wieder. Es ermöglicht somit eine Momentaufnahme zur Laufzeit des Programms darzustellen, in der die Attribute der Klassen mit konkreten Werten belegt sind.

---

[31] Weitere Details zu Serialisierung und Deserialisierung sind aus entsprechenden Literaturquellen zu entnehmen (z.B. [30] oder [32]).

Diese beiden Sichten auf den Stand-Alone-Modus ermöglichen es, die komplette Anwendungsarchitektur sowie das resultierende Programmverhalten exakt zu beschreiben und übersichtlich darzustellen.

Abbildung 5-11 zeigt die wesentlichen Klassen des PM-Moduls der P2-Plantafel sowie deren Beziehungen durch Vererbung, Referenzierung und Komposition. In Anlehnung an die Unified Modeling Language (UML) lautet die Notation für Attribute "Attribut : Datentyp", während Methoden mit einer Doppelklammer ( ) gekennzeichnet werden. Vererbung wird mit gestrichelten, Referenzierung mit durchgehenden und Komposition mit gepunkteten Pfeilen dargestellt.
Weiterhin werden zur Verbesserung der Lesbarkeit Präfixe für die Benennung von Klassen und Variablen eingeführt. "C" kennzeichnet Klassen, "CView" Ansichtsklassen, "l" Listen, "d" Kalenderdaten, "a" Arrays und "f" Flags. Außerdem werden die Namen von Klassen, die aus dem MFC-Framework übernommen sind grau hinterlegt, während die Namen von Klassen die speziell für die P2P entwickelt sind schwarz dargestellt werden.
Zudem sind aus Gründen der Übersichtlichkeit Zugriffsfunktionen, Konstruktoren, Systemfunktionen sowie triviale Funktionen nicht explizit aufgeführt.

Abbildung 5-11 zeigt das Klassendiagramm für das komplette PM-Modul. Im Anschluss werden zunächst die Klassen der Logikschicht behandelt. Die Klassen der Präsentationsschicht werden im Rahmen des darauffolgenden Unterkapitels vorgestellt (vgl. Kapitel 5.5.3).

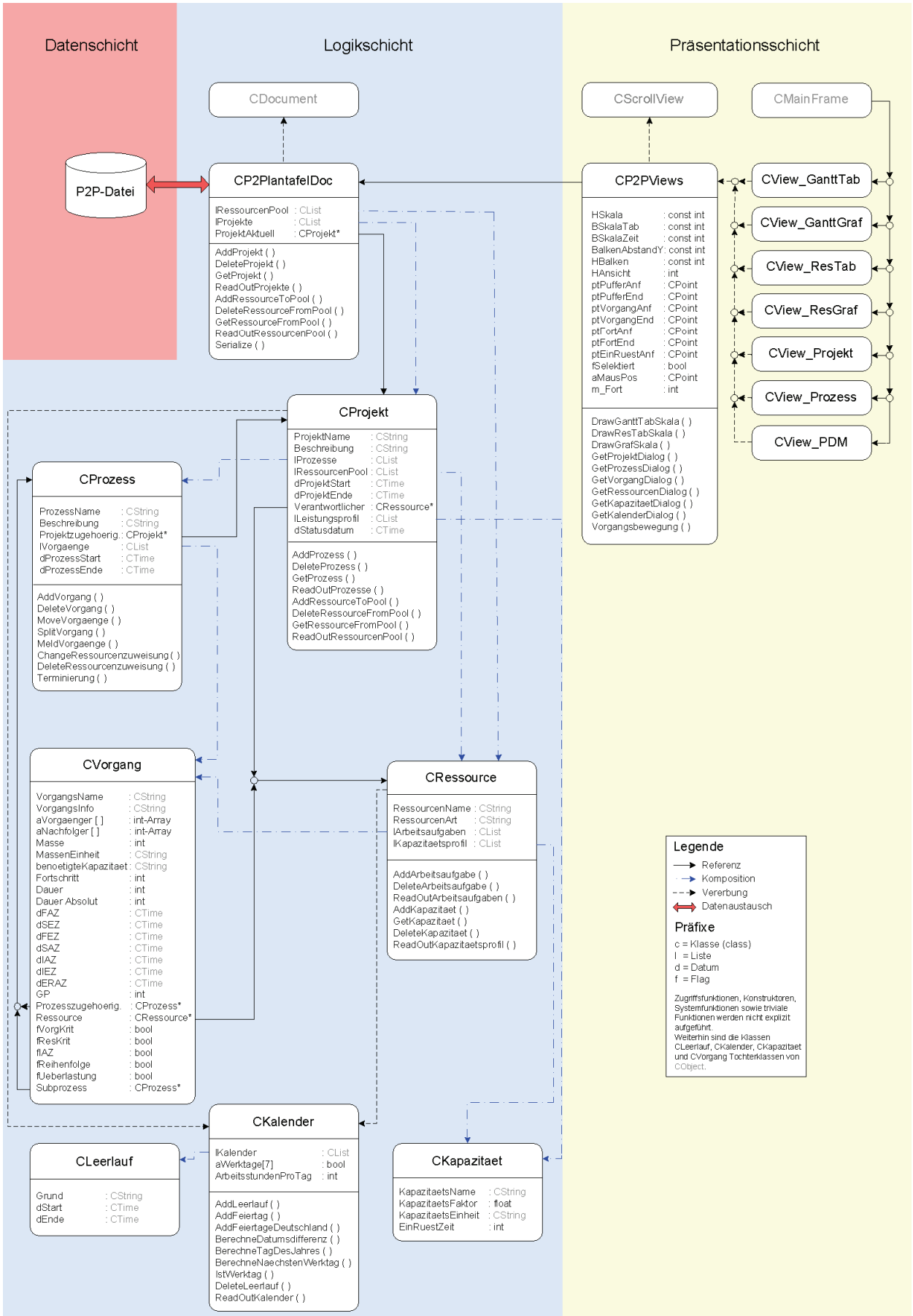


Abbildung 5-11: Klassendiagramm des PM-Moduls

Die Klassen der Präsentationsschicht nehmen im Wesentlichen drei verschiedene Funktionen wahr. Während die Klassen ***CProjekt***, ***CProzess***, ***CVorgang*** und ***CRessource*** der Abbildung eines einzelnen Projektes dienen, stellt die Klasse ***CP2PlantafelDoc*** Funktionalitäten für das Multiprojektmanagement sowie die Schnittstelle zu den Ansichtsklassen zur Verfügung. Außerdem stellen die Klassen ***CLeerlauf***, ***CKalender*** und ***CKapazitaet*** grundlegende Objekte und Methoden für die zuvor genannten Klassen bereit.
Nachfolgend werden die einzelnen Klassen einer eingehenden Betrachtung unterzogen. Dabei wird vorwiegend auf die zentralen Eigenschaften der jeweiligen Klassen eingegangen.

CTime, CKalender, CLeerlauf,

Grundvoraussetzung für die Planung von Projekten ist ein zeitlicher Bezugsrahmen. Dieser wird mittels der Klassen CTime, CKalender und CLeerlauf abgebildet.

Die elementare Zeiterfassung erfolgt auf Grundlage der Klasse **CTime** aus dem MFC-Framework. Diese Klasse kapselt eine Datenstruktur zur Erfassung von Zeitpunkten, die sich aus den Variablen Tag, Monat und Jahr sowie Stunde, Minute und Sekunde zusammensetzt. Weiterhin verfügt diese Klasse über eine Reihe von Operatoren, welche die Durchführung von Rechen- und Vergleichsoperationen ermöglichen.

Die Erfassung von Zeitintervallen in denen Ressourcen nicht für die Bearbeitung von Vorgängen zur Verfügung stehen, erfolgt primär mit Hilfe der Klasse **CLeerlauf**. Diese besteht aus einem Start- und einem Endzeitpunkt sowie einer Zeichenkette welche den Grund für den Leerlauf erfasst.

Die Summe der anfallenden Leerlaufzeiten wird in einem Kalender zusammengefasst. Die Klasse **CKalender** beinhaltet eine dynamische Datenstruktur, welche eine Komposition aus den anfallenden Leerlaufzeiten bildet. Werktage an denen generell Leerlaufzeiten anliegen, werden dabei zur Reduzierung der benötigten Datenmenge nicht in Form von Leerläufen, sondern als arbeitsfreie Werktage erfasst. Dadurch kann beispielsweise ohne die Verwendung von CLeerlauf modelliert werden, dass die Ressource "Meier" nur montags bis freitags zur Verfügung steht.
Da die Arbeitsleistung einer Ressource neben den Arbeitstagen von der Anzahl der Arbeitsstunden abhängt, erfolgt in CKalender zudem die Erfassung der Anzahl von Arbeitsstunden pro Tag.

Wie einleitend bereits erläutert wurde, wird bei der P2-Plantafel zwischen Ressourcen- und Projektkalender unterschieden. Dies dient sowohl der Übersichtlichkeit als auch der

Reduzierung der Datenmenge. Zeitintervalle, an denen alle Ressourcen des Projektes generell nicht zur Verfügung stehen, werden einmal an zentraler Stelle erfasst (Projektkalender), so dass bei den einzelnen Ressourcen jeweils nur die individuellen Ausfallzeiten berücksichtigt werden müssen. Die Klasse CKalender dient aus diesem Grund sowohl als Basisklasse für die Klasse der Ressourcen (CRessource) als auch für die des Projektes (CProjekt).

Weiterhin umfasst **CKalender** eine Reihe von Methoden, die für die Berechnungsalgorithmen des Projektes erforderlich sind:

- AddFeiertageDeutschland: Fügt dem Projektkalender automatisch alle deutschlandweiten Feiertage für ein bestimmtes Jahr hinzu
- AddFeiertag: Fügt dem Projektkalender manuell einen benutzerdefinierten Feiertag hinzu (notwendig für regionale Feiertage)
- BerechneDatumsdifferenz: Berechnet die Länge des Zeitintervalls zwischen zwei Zeitpunkten
- BerechneTagDesJahres: Bestimmt den Tag des Jahres (z.B.: 2. Februar = Tag 33)
- IstWerktag: Prüft, ob ein Datum auf einem Werktag liegt
- BerechneNaechstenWerktag: Berechnet von einem Datum den nächsten Werktag

CKapazitaet

Die Klasse CKapazitaet dient der Abbildung des Leistungsvermögens von Ressourcen. Jede Ressource besitzt innerhalb einer Planungsperiode eine endliche Kapazität [2]. Um ein standardisiertes Berechnungsverfahren für alle Arten von humanen und technischen Ressourcen zu konzeptionieren, ist für alle Ressourcenarten eine einheitliche Klassifizierung der Kapazität erforderlich. Die Kapazität setzt sich daher aus drei verschiedenen Attributen zusammen:

- **Name der Kapazität**

  Der Name der Kapazität beschreibt, um welche Art von Kapazität es sich handelt. Bei humanen Ressourcen wird das Leistungsvermögen nach Fähigkeiten aufgeschlüsselt, während sich technischen Ressourcen durch Funktionen definieren. Die Summe aller Kapazitäten einer Ressource bildet ihr Kapazitätsprofil.

- **Größe der Kapazität**

  Die Größe der Kapazität quantifiziert das Leistungsvermögen einer Ressource für eine bestimmte Art von Kapazität. Sie setzt sich zusammen aus dem Kapazitätswert und der Maßeinheit der Kapazität (Kapazitätseinheit). Die Maßeinheit der Kapazität besitzt das Format 'abstrakte Masseneinheit pro Zeiteinheit' (z.B. 'Stück pro Stunde').

Bei humanen Ressourcen ist die Größe der Kapazität als ein Mess- oder Erfahrungswert für den Grad des Könnens auf einem bestimmten Gebiet zu verstehen, während bei technischen Ressourcen auf Grundlage von technischen Kennwerten oftmals ein absoluter Wert ermittelt werden kann.

• **Einarbeitungs- bzw. Rüstzeit**

Die Einarbeitungs- bzw. Rüstzeit erfasst die Zeit, die eine Ressource zur Vorbereitung von Arbeiten auf einem bestimmten Gebiet benötigt. Im weiteren Verlauf dieser Arbeit wird dieses Attribut auch als Einrüstzeit bezeichnet.

Die folgende Abbildung 5-12 zeigt die Kapazitätsprofile für unterschiedliche humane und technische Ressourcen. Bei einigen Kapazitäten wurde zugunsten einer vereinfachten Darstellung auf die Konkretisierung einer Einheit verzichtet. In diesem Fall ist als Einheit der Platzhalter E eingesetzt. Die nachfolgenden Beispiele sollen den Nutzen des neuartigen Konzeptes für die optimierte Erfassung von Ressourceneigenschaften verdeutlichen.

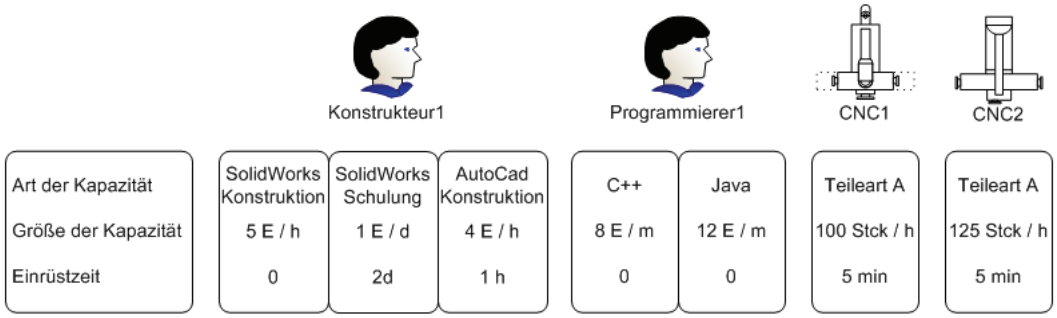


Abbildung 5-12: Kapazitätsprofile

Aus den abgebildeten Kapazitätsprofilen ist ersichtlich, dass ein Vorgang der Programmierkenntnisse in C++ erfordert, aufgrund der vorliegenden Qualifikationen, zwar von der Ressource Programmierer1 abgewickelt werden kann, allerdings nicht von Konstrukteur1. Durch die Unterscheidung verschiedener Kapazitätsarten wird somit sichergestellt, dass die Fähigkeiten bzw. Funktionen der Ressourcen den Anforderungen der zugeteilten Vorgänge entsprechen.

Währenddessen ermöglicht die Größe der Kapazität eine Quantifizierung des Einflusses der Ressourcenzuteilung auf die Vorgangsdauer. Während die technische Ressource CNC1 zum Beispiel von der Teileart A 100 Stück pro Stunde fertigen kann, stellt die Ressource CNC2 in jeder Stunde 125 Stück her. Dadurch kann die Produktion von einer vorgegebenen Anzahl von Werkstücken der Teileart A von der Ressource CNC2 25% schneller abgeschlossen werden, als von der Ressource CNC1.

Weiterhin ist bei der Ressource Konstukteur1 zu berücksichtigen, dass er, bevor er eine Schulung in dem CAD-System SolidWorks beginnen kann, zur Einarbeitung in die Schulungsunterlagen einen Vorlauf von 2 Tagen benötigt. Durch Erfassung der Einrüstzeit kann dieser Aspekt in die Zeitplanung des Projektes einbezogen werden.

Aus diesen einfachen Beispielen wird bereits deutlich, dass die Erfassung von Kapazitäten für die Ergebnisgüte der Projektplanung von essenzieller Bedeutung ist. Bei der Planung von Projekten mit herkömmlichen PM-Systemen wird das Leistungsvermögen von Ressourcen in der Regel nicht in dieser Granularität aufgeschlüsselt. Im Gegensatz dazu ist die genaue Auswertung des Leistungsvermögens von Ressourcen ein integraler Bestandteil des P2P-Konzeptes. Dies stellt eine beachtliche Optimierung der EDV-Unterstützung der Produktentwicklung dar.

CProzess

Die Hauptaufgabe der Klasse CProzess besteht darin, sämtliche relevanten Daten von einem Arbeitsablauf zu erfassen und Methoden zur Manipulation dieser Daten sowie Berechnungsverfahren bereitzustellen. Wie eingehend bereits erläutert wurde, ist ein Arbeitsablauf geprägt durch die zu bearbeitenden Vorgänge und die zur Verfügung stehenden Ressourcen. Während Vorgänge einem konkreten Prozess zuzuordnen sind, werden Ressourcen auf Projektebene verwaltet. In der Klasse CProzess erfolgt daher die Administration der Vorgänge sowie die Ressourcenzuweisung, während die zur Verfügung stehenden Ressourcen von hierarchisch höheren Klassen verwaltet werden (CProjekt und CP2PlantafelDoc).

Ein wesentlicher Aspekt des Konzeptes für das PM-Modul besteht in der **Speicherverwaltung** der Vorgänge und Ressourcen. Da die Anzahl dieser Elemente während der Projektplanung variieren kann, ist die Verwendung einer dynamischen Datenstruktur erforderlich. Weiterhin muss berücksichtigt werden, dass die für das Projektmanagement erforderlichen Berechnungen extrem rechenintensiv sind und einen schnellen Zugriff auf die einzelnen Elemente benötigen. Aus diesen Gründen wird in CProzess für die Erfassung der zum Projekt gehörenden Vorgänge eine doppelt verkette Liste[32] verwendet. Aus den gleichen

[32] Doppelt verkettete Liste: *dynamische Datenstruktur, die eine Speicherung von einer im Vorhinein nicht bestimmten Anzahl von miteinander in Beziehung stehenden Werten einfacher oder zusammengesetzter Datentypen erlauben.* Jedes Listenelement ist durch einen Zeiger mit dem vorausgehenden und dem nachfolgenden Listenelement verknüpft [6].

Gründen wird diese Datenstruktur für sämtliche weiteren Kompositionen innerhalb des PM-Moduls eingesetzt.

Die P2-Plantafel unterstützt sowohl die parallele Bearbeitung von voneinander abhängigen Prozessen als auch die parallele Bearbeitung von untereinander abhängigen Projekten (Mulitprozess- und Multiprojektmanagement). Aus diesem Grund ist es erforderlich, dass jeder Prozess das zugehörige Projekt referenziert.

Der Planungsverantwortliche ist durch diesen Aspekt des P2P-Konzeptes in der Lage, die Wechselwirkung zwischen unterschiedlichen Prozessen im Kontext ihrer Projekte gezielt auszuwerten.

CRessource

Die verfügbaren Ressourcen werden durch Objekte der Klasse CRessource repräsentiert. Die wichtigste Eigenschaft einer jeden Ressource ist ihr Kapazitätsprofil. Dabei handelt es sich um eine Aufschlüsselung der Qualifikation einer Ressource für konkrete Arbeitsaufgaben. Das Kapazitätsprofil wird als Komposition beliebig vieler Objekte der Klasse CKapazitaet in Form einer dynamischen Datenstruktur realisiert.

Weiterhin referenziert jede Ressource in einer zweiten dynamischen Datenstruktur sämtliche zugeteilten Arbeitsaufgaben (Vorgänge). Dies ist erforderlich, da das PM-Modul die parallele Bearbeitung verschiedener Prozesse und Projekte ermöglicht. Daher muss die Information verfügbar sein, welche Ressourcen für die Bearbeitung von Vorgängen aus anderen Prozessen eingeteilt sind. Dabei werden sowohl Prozesse aus dem aktiven Projekt als auch Prozesse aus anderen Projekten berücksichtigt.

CVorgang

Die Ebene der Vorgänge enthält Vorgaben für die im Zuge eines Prozesses durchzuführenden Arbeitsschritte. Die Angabe der Vorgangseigenschaften bildet somit die Grundlage für die Planung von Prozessen.

Die **Prozess-Struktur** resultiert aus den Beziehungen zwischen den einzelnen Vorgängen. Um die Verwaltung der Vorgangsbeziehungen möglichst transparent zu gestalten, werden von jedem Vorgang die Positionsnummern der unmittelbar vorausgehenden Vorgänge (Vorgänger) sowie der unmittelbar nachfolgenden Vorgänge (Nachfolger) erfasst. Um eine variable Anzahl von Vorgangsbeziehungen zu unterstützen, werden die angegebenen Vorgänger und Nachfolger in einer dynamische Datenstruktur abgelegt. Weiterhin wird durch die Implementierung entsprechender Methoden ein redundanzfreier Datenbestand gewährleistet. Das bedeutet, dass bei Angaben eines Nachfolgers von Vorgang A, beispielsweise bei dem referenzierten Vorgang automatisch Vorgang A als Vorgänger eingesetzt wird.

Da die P2-Plantafel neben dem Multiprojektmanagement auch das Multiprozessmanagement unterstütz, ist es erforderlich, dass jeder Vorgang den zugehörigen Prozess referenziert.

Die **Dauer** eines Vorganges ist eine variable Größe, die von dem vorliegenden Planungsszenario abhängt. Basis für die Berechnung der Vorgangsdauer ist die Information, welche Ressource den Vorgang bearbeitet sowie die Masse des Vorganges.

Bei der **Masse** handelt es sich um eine abstrakte Größe, die den Arbeitsaufwand für einen Vorgang quantifiziert. Die Einheit der Masse ist vorgangsspezifisch und kann anhand des Attributes MassenEinheit frei definiert werden. Dadurch ist sichergestellt, dass in dem Plantafel-Konzept sämtliche Arten von Vorgängen berücksichtigt sind.
Der Arbeitsaufwand für eine Teilefertigung kann beispielsweise über die MassenEinheit 'Stückzahl' definiert werden. Der Aufwand für eine Programmieraufgabe könnte hingegen über die Anzahl der zu implementierenden Klassen (MassenEinheit 'Klassen') oder auch durch eine Abschätzung des Programmumfanges (MassenEinheit 'Programmzeilen') determiniert werden.
Für Vorgänge deren Arbeitsaufwand nicht in Mengenbegriffen ausgedrückt werden kann, ist eine Zeitabschätzung durchzuführen (MassenEinheit 'Tage', 'Stunden' u.ä.). Dabei empfiehlt es sich als Masse die Zeit zu veranschlagen, die eine durchschnittliche Ressource für die Bearbeitung des betreffenden Vorgangs schätzungsweise benötigt.

Um die unterschiedlichen Qualifikationen der verfügbaren Ressourcen in die Projektplanung einzubeziehen, erfolgt in CVorgang zudem die Angabe der Kapazität, welche für die Bearbeitung eines Vorgangs erforderlich ist. Ein Abgleich der **benötigten Kapazität** mit der bei der zugeteilten Ressource vorhanden Kapazitäten, ermöglicht eine Bestimmung der Vorgangsdauer in Abhängigkeit von der Ressourcenqualifikation.

Die aus diesen Angaben berechnete Dauer ist allerdings eine Scheingröße, da sie die Länge eines Vorganges in Arbeitstagen quantifiziert. Daher ist die Einführung einer weiteren Größe erforderlich, welche die Länge eines Vorgangs in Kalendertagen erfasst. Diese Größe wird im Folgenden als **absolute Dauer** bezeichnet. Bei unveränderten Randbedingungen, d.h. unveränderter Masse, benötigter Kapazität und zugeteilter Ressource, stellt die Vorgangsdauer eine konstante Größe dar, während die absolute Dauer von der zeitlichen Lage des Vorganges abhängt. Wenn ein Vorgang mit einer Dauer von 4 Tagen beispielsweise auf ein Wochenende fällt und die zugeteilte Ressource an diesen Tagen frei hat, beträgt die absolute Dauer nicht 4 sondern 6 Tage.

Das im Rahmen der Konzeption der P2-Plantafel entwickelte Berechnungsverfahren für die Bestimmung von Vorgangsdauern und Vorgangsterminen wird ab Seite 80 einer ausführlichen Betrachtung unterzogen. An dieser Stelle soll zunächst auf die **Randbedingungen** bei der Vorgangsmodellierung eingegangen werden.

Laut Definition wird ein Vorgang ohne Unterbrechung abgewickelt. Wenn dies nicht bewerkstelligt werden kann, ist der Vorgang in Teilvorgänge zu zerlegen. Weiterhin wird vorausgesetzt, dass die Bearbeitung eines Vorganges von genau einer Ressource durchgeführt wird und der Ressourceneinsatz während der gesamten Vorgangsdauer konstant bleibt. Dadurch kann eine proportionale Beziehung zwischen der Vorgangsdauer und dem Arbeitsfortschritt vorausgesetzt werden. Das Controlling des Projektes kann infolgedessen mittels eines Soll-Ist-Vergleichs des Vorgangsfortschrittes und des Statusdatums des Projektes realisiert werden (vgl. CProjekt).

Um die Projektleitung über Verspätungen, Restriktionsverletzungen und ähnliches zu informieren, enthält die Klasse CVorgang zudem eine Reihe von Flags[33], die entsprechende Inkonsistenzen kennzeichnen.

CProjekt

Die Klasse CProjekt bildet die oberste Hierarchieebene eines Planungsvorhabens. Sie verwaltet sowohl die einzelnen Prozesse des Planungsvorhabens als auch die zur Verfügung stehenden Ressourcen.

Weiterhin kann mittels Referenzierung einer Ressource aus dem Ressourcenpool ein Projektverantwortlicher bestimmt werden. Der Schreibzugriff auf das Projekt kann dadurch auf eine bestimmte Person beschränkt werden.

Ferner wird von der Klasse CProjekt das sogenannte Leistungsprofil gebildet. Diese Datenstruktur quantifiziert das durchschnittliche Leistungsvermögen der Ressourcen des Projektes und ist für die überschlägige Berechnung der Projektdauer erforderlich. Sie setzt sich aus exakt so vielen Objekten der Klasse CKapazitaet zusammen, wie unterschiedliche Kapazitätsarten im Projekt vorhanden sind. Die Kapazitätsgröße und die Einrüstzeit der einzelnen Kapazitätsarten des Leistungsprofils errechnen sich aus dem Durchschnitt der im Ressourcenpool vorhandenen Kapazitäten.

CP2PlantafelDoc

Während CProjekt und CProzess im Zentrum der Abbildung eines konkreten Einzelprojektes stehen, ist die Klasse CP2PlantafelDoc als übergeordnete Instanz zum Handling aller vorliegenden Projekte zu sehen. In der hierarchischen Struktur der Logikschicht des PM-Moduls steht die Dokumentenklasse CP2PlantafelDoc daher an oberster Stelle.

[33] Flag: Variable, die zwei verschiedener Zustände annehmen kann

CP2PlantafelDoc stellt die Funktionalitäten für die Verwaltung und Speicherung aller Plantafel-Projekte zur Verfügung. Sie enthält daher eine Datenstruktur zur Erfassung von Objekten des Typs CProjekt sowie Funktionen zum Einfügen, Auslesen und Löschen von Projekt-Objekten.
Außerdem wird die Gesamtheit der zur Verfügung stehenden Ressourcen von CP2PlantafelDoc verwaltet. Der Ressourcenpool von Projekten stellt eine Teilmenge dieser Ressourcen dar. Wenn eine vollständige Integration der P2-Plantafel in die IT-Infrastruktur einer Firma erfolgt ist, beinhaltet der Ressourcenpool von CP2PlantafelDoc sämtliche Mitarbeiter der Firma, während der Pool von CProjekt die Mitarbeiter des Projektteams umfasst.

Weiterhin bildet CP2PlantafelDoc die Schnittstelle zu den Ansichtsklassen der Präsentationsschicht und der P2P-Datei der Datenschicht. Die dazu notwendigen Funktionen werden von dem MFC-Framework bereitgestellt. Aus diesem Grund werden Sie an dieser Stelle nicht weiter ausgeführt. Es sei lediglich darauf hingewiesen, dass die Speicherung des Plantafel-Dokumentes durch Überschreiben der Serialize-Funktion des MFC-Frameworks realisiert wird. Da diese Funktion alle Daten des Dokumentes sequentiell in eine Datei schreibt bzw. aus einer Datei liest, müssen alle von CP2PlantafelDoc verwendeten Datenelemente serialisierbar sein. Daher ist es erforderlich, dass alle Klassen des PM-Moduls mittels direkter oder indirekter Vererbung von der MFC-Klasse CObject abstammen.

Eine weitere Aufgabe von CP2PlantafelDoc besteht darin, in Bearbeitung befindliche Objekte zu referenzieren. Dadurch ist für die Ansichtsklassen der Präsentationsschicht jederzeit ersichtlich welche Objekte selektiert sind und sich infolgedessen in Bearbeitung befinden. Dies umfasst sowohl das aktive Projekt als auch den aktiven Prozess, den aktiven Vorgang, die aktive Ressource und die aktive Kapazität. Aus Gründen der Übersichtlichkeit wird in dem gezeigten Klassendiagramm allerdings lediglich das aktuelle Projekt referenziert (vgl. Abbildung 5-11)

Von den unterschiedlichen Klassen des PM-Moduls werden zur Laufzeit der Plantafel konkrete Objekte instanziert. Das auf der Folgeseite abgebildete Objektdiagramm stellt die Konstellation dieser Objekte für eine fiktive Planungssituation dar.

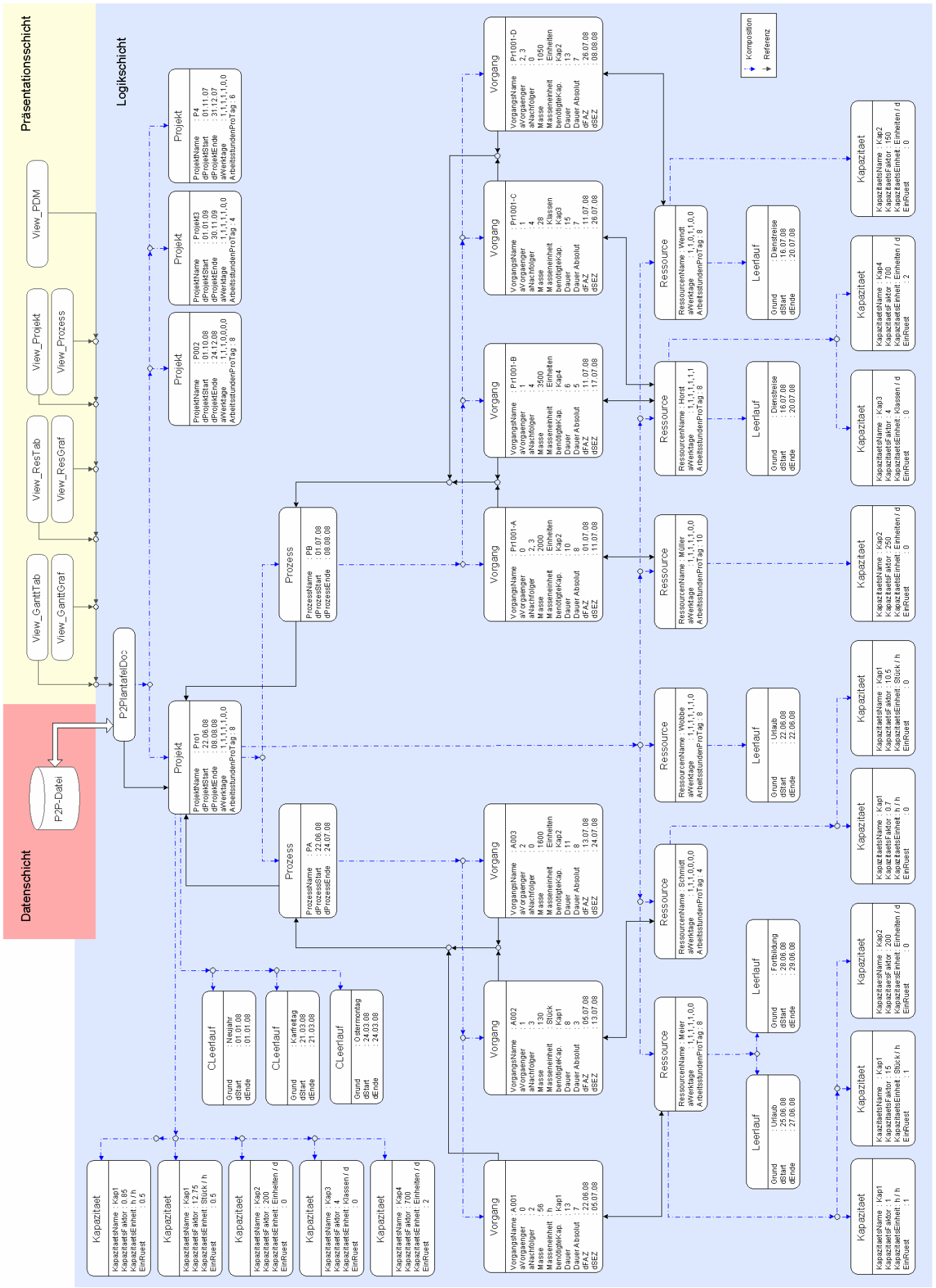


Abbildung 5-13: Objektdiagramm des PM-Moduls

Aus Gründen der Übersichtlichkeit sind in dem beispielhaften Objektdiagramm lediglich ein Projekt mit einem schlanken Ressourcenpool und Prozesse mit einer geringen Anzahl von Vorgängen dargestellt. Weiterhin sind die Kapazitätsprofile der Ressourcen stark vereinfacht und der Umfang der Projekt- und Ressourcenkalender soweit wie möglich reduziert. Zudem sind in dem Objektdiagramm aus Platzgründen lediglich die zentralen Attribute der einzelnen Objekte abgebildet und sämtliche Methoden vernachlässigt.

Wie zuvor bereits erläutert wurde, agiert das Objekt **P2PlantafelDoc** als oberste Instanz der Logikschicht. Es dient als Container für alle vorliegenden Projekte und fungiert zudem als Schnittstelle zu der Daten- und der Präsentationsschicht.
P2PlantafelDoc steht mit den verwalteten Projekten in einer Kompositionsbeziehung. Die Komposition wird rechnerintern mittels einer dynamischen Datenstruktur realisiert, so dass von der Plantafel beliebig viele Projekte administriert werden können.
Das in Bearbeitung befindliche Projekt wird von P2PlantafelDoc referenziert, so dass aus der Präsentationsschicht heraus ein direkter Zugriff auf das aktive Projekt möglich ist. Die Bearbeitung von einem in dem aktiven Projekt selektierten Prozess, Vorgang, Ressource, Leerlauf oder Kapazität wird analog realisiert. Diese Referenzierungen werden aus Gründen der Übersichtlichkeit in dem Objektdiagramm nicht explizit aufgeführt. Weiterhin wird aus Platzgründen lediglich die Struktur eines einzelnen Projektes detailliert dargestellt. Dabei handelt es sich um das Projekt 'Pro1'.

Das **Projekt** 'Pro1' steht im Mittelpunkt des beispielhaften Objektdiagramms. Da die Klasse CProjekt eine Tochterklasse von CKalender ist, beinhaltet jedes Projektobjekt einen Projektkalender. Dieser enthält neben den Attributen für die Rahmenarbeitszeiten (Werktage und ArbeitsstundenProTag) die Erfassung aller arbeitsfreien Tage in Form einer Komposition aus Leerlauf-Objekten. Dem Projektkalender können neben den deutschlandweiten Feiertagen des Projektjahres bzw. der Projektjahre beliebig viele benutzerdefinierte Leerläufe hinzugefügt werden. Das vorliegende Objektdiagramm beschränkt sich aus Platzgründen auf die ersten Leerläufe des Projektkalenders.
Neben dem Projektkalender verwaltet jedes Projekt ein Leistungsprofil. Diese Komposition aus Kapazitätsobjekten repräsentiert das durchschnittliche Leistungsvermögen aller Ressourcen des Projektes und ermöglicht eine Approximation der Dauern von Vorgängen, denen keine passende Ressource zugeteilt wurde. Für den Vorgang A003 kann ausgehend von den Rahmenarbeitszeiten und dem Leistungsprofil des Projektes, beispielsweise auch ohne Zuteilung einer Ressource, eine Vorgangsdauer bestimmt werden.
Zusätzlich zu den dargelegten Kompositionen umfasst jedes Projekt-Objekt eine Vielzahl von Attributen, welche die Basisinformationen des Projektes abbilden. Für das Projekt Pro1 ist

beispielsweise ersichtlich, dass unter den vorliegenden Randbedingungen bei einem Projektstart am 22.06.08 das Projektende für den 08.08.08 terminiert ist.

Die Arbeitsabläufe, die im Zuge eines Projektes anfallen, werden durch Objekte vom Typ **Prozess** verwaltet. Jeder Prozess setzt sich aus einer Reihe von Arbeitsaufgaben zusammen. Diese werden durch Objekte vom Typ **Vorgang** repräsentiert, die zu dem Prozess-Objekt in einer Kompositionsbeziehung stehen. Für die Implementierung des Multiprozessmanagements ist es erforderlich, dass jeder Vorgang den zugehörigen Prozess referenziert.
Die hierarchische Struktur eines Prozesses wird durch Angabe der Vorgänger und Nachfolger der einzelnen Vorgänge erfasst. Da die Datenstruktur, in der die Vorgänge gespeichert werden, über eine Indizierung verfügt, wird diese Information systemintern als Integer-Wert verarbeitet.
Zur vollständigen Abbildung der komplexen Zusammenhänge zwischen den einzelnen Vorgängen eines Prozesses, ist neben der Prozessstruktur die Angabe einer Vielzahl weiterer Informationen erforderlich. Dabei handelt es sich im Wesentlichen um Termine und Dauern, die in Form von entsprechenden Vorgang-Attributen erfasst werden.

Für die Berechnung dieser Größen sind die den Vorgängen zugeteilten **Ressourcen** von zentraler Bedeutung. Jedes Projekt besitzt eine endliche Anzahl von Ressourcen, die für die Bearbeitung der einzelnen Vorgänge eingesetzt werden. Von jedem Vorgang wird die ihm zugeteilte Ressource referenziert. Weiterhin wird von dem Projekt-Objekt in Form einer Komposition auf sämtliche in dem Projekt verfügbaren Ressourcen verwiesen (Ressourcenpool). Dabei kann eine Ressource zu mehreren Projekten gehören. Um terminliche Überschneidungen zu erfassen, ist es erforderlich, dass von jeder Ressource sämtliche zugeteilten Vorgänge referenziert werden. Bei der Berechnung der Termine für ein aktuelles Projekt kann somit die Belegung von Ressourcen durch parallel ablaufende Prozesse und Projekte berücksichtigt werden.

Weiterhin verfügt jede Ressource über einen Ressourcenkalender, der es ermöglicht arbeitsfreie Zeiten in die Terminberechnung einzubeziehen. Analog zu dem Projektkalender setzt sich dieser aus einer Komposition von **Leerlauf**-Objekten sowie entsprechenden Attributen zusammen.

Die Leistungsfähigkeit der unterschiedlichen Ressourcen wird mittels Objekten vom Typ **Kapazität** erfasst. Da das Leistungsspektrum der vorhandenen Ressourcen in der Regel eine große Streuung aufweist, handelt es sich auch dabei um eine Kompositionsbeziehung. Von einem Ressourcen-Objekt kann folglich auf beliebig viele Kapazität-Objekte verwiesen werden.

Die in diesem Objektdiagramm aufgezeigte Darstellung eines rudimentären Projektes zeigt bereits deutlich, dass die Masse an Informationen zu einem Projekt schnell eine Komplexität erreicht, die ohne eine geeignete Softwareunterstützung nicht mehr effizient zu bewältigen ist.

Das für das PM-Modul konzipierte Datenmodell ermöglicht auf Grundlage der vorgestellten Objekte und der entwickelten Verknüpfungen eine flexible Abbildung von beliebig komplexen Projekten. Das entwickelte Datenmodell gewährleistet somit, dass die Fülle an Informationen zu verschiedenen Projekten organisiert verwaltet werden.

Die Projektmanagement-Funktionalität des PM-Moduls wird durch die Methoden der Logikschicht bereit gestellt. Wie das Klassendiagramm auf Seite 68 verdeutlicht, dienen nahezu alle Methoden der Logikschicht dazu, die Masse an vorliegenden Informationen zu administrieren. Auf Funktionsweise dieser trivialen Methoden soll an dieser Stelle nicht detailliert eingegangen werden. An dieser Stelle wird sich stattdessen auf die zentrale Methode der Logikschicht konzentriert. Die Methode ***Terminierung***[34] dient dazu, auf Grundlage der aktuellen Informationen zum Projekt die zeitliche Lage sämtlicher Vorgänge und deren Dauer zu berechnen.

Die Grundlage für das Berechnungsverfahren zur Terminierung von P2P-Projekten bildet die Methode des kritischen Pfades (Critical Path Method) aus der Netzplantechnik[35]. Dieses deterministische Berechnungsverfahren führt auf Basis eines volldefinierten Netzplans[36] eine Zeitanalyse durch und kennzeichnet kritische Planelemente (kritischer Pfad). Es sind dabei der Netzplan und die Dauern der Vorgänge fest vorgegeben und es werden die Fristen bzw. die Termine der einzelnen Vorgänge berechnet.

Abbildung 5-14 zeigt den Ablauf der CPM-Berechnung sowie die dazugehörigen Formeln.

---

[34] Terminierung: einen Termin, eine Frist für etwas bestimmen [43]

[35] Netzplantechnik: *Alle Verfahren zur Analyse, Planung, Steuerung und Überwachung von Abläufen auf der Grundlage der Graphentheorie, wobei Zeit, Kosten, Einsatzmittel bzw. Ressourcen und weitere Einflussgrößen berücksichtigt werden können*. [40]

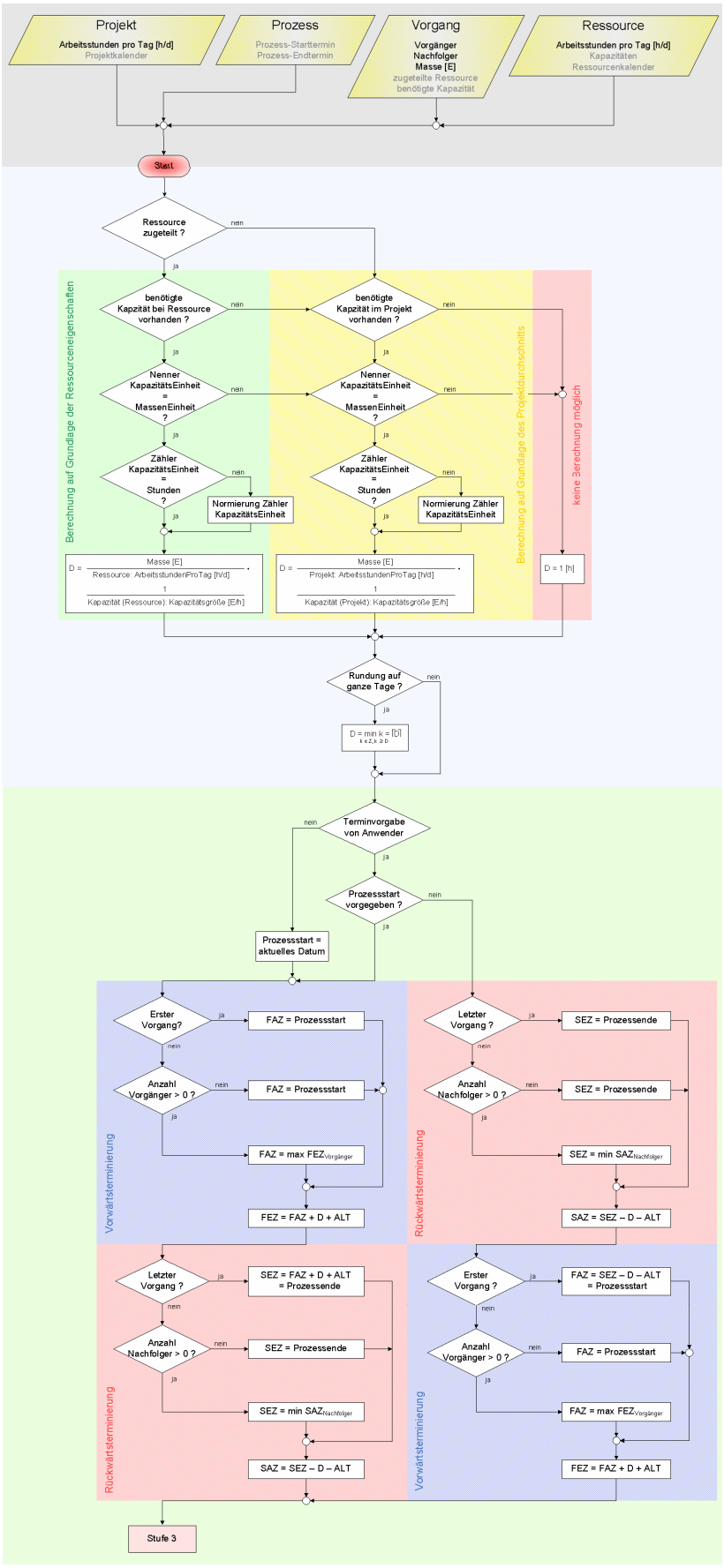
[36] Ein Netzplan gibt den Projektablauf durch Vernetzung der einzelnen Vorgänge entsprechend ihrer logischen und technologischen Abhängigkeiten wider.

Vorwärtsrechnung: $FAZ_j = \max(FEZ_i)$ $FEZ_j = FAZ_j + D_j$ Rückwärtsrechnung: $SEZ_j = \min(SAZ_k)$ $SAZ_j = SEZ_j - D_j$ Sonderfälle: $FAZ_1 = 0$ $SEZ_n = FEZ_n$	Pufferberechnung: $GP_j = SAZ_j - FAZ_j$ $FP_j = \min(FAZ_k) - FEZ_j$ $FRP_j = SAZ_j - \max(SEZ_i)$ $UP_j = \max(\min(FAZ_k) - \max(SEZ_i) - D)$ kritischer Pfad: $GP_j = 0$	• FAZ, FEZ, SEZ, SAZ: Anfangs- und Endzeitpunkte eines Vorganges (jeweils die frühest- und spätestmögliche Lage) • D: Vorgangsdauer • GP, FP, FRP, UP: verschiedene Pufferzeiten • Indizes: j = aktueller Vorgang i = Vorgänger k = Nachfolger 1 = erster Vorgang (kein Vorgänger) n = letzter Vorgang (kein Nachfolger)

Abbildung 5-14: CPM-Berechnungsablauf [19]

Der in Abbildung 5-14 gezeigte Berechnungsablauf ist für die Terminierung der P2-Plantafel unzureichend, weil die Berechnung aller Termine auf Grundlage von Werktagen erfolgt und die Ressourcenzuweisung nicht in die Berechnung einbezogen wird. Infolgedessen bleibt unberücksichtigt, dass Kapazität, Arbeitsstunden und Leerlaufzeiten der zugeteilten Ressourcen erheblichen Einfluss auf die Dauern der einzelnen Vorgänge ausüben.
Weiterhin beruht das CPM-Berechnungsverfahren auf der Prämisse, dass sich das Projekt, wie es im Projektmanagement üblich ist, aus einem einzelnen Arbeitsablauf zusammensetzt. Es wird also kein Multiprozessmanagement unterstützt.

Aus diesen Gründen ist im Sinne einer Optimierung der Produktentwicklung für die P2-Plantafel die Konzipierung eines neuen Berechnungsverfahrens erforderlich. Abbildung 5-15 zeigt die 5 Stufen des für die Terminierung von P2P-Projekten entwickelten Berechnungsverfahrens (rechts) sowie das dazugehörige Flussdiagramm (links). Das Flussdiagramm visualisiert den Ablauf bei der Terminierung eines einzelnen Vorganges und ist nacheinander von allen Vorgängen sämtlicher Prozesse des Projektes zu durchlaufen. Der Projektstarttermin ergibt sich im Anschluss an diese Berechnungen aus dem Starttermin des frühesten Prozesses. Währendessen ist der Projektendtermin durch den Endtermin des letzten Prozesses determiniert.



Eingangsgrößen

1. Berechnung der Vorgangsdauer in Abhängigkeit von der Ressourcenkapazität

2. Berechnung der Vorgangsgrenzen durch Vorwärts- & Rückwärts-terminierung

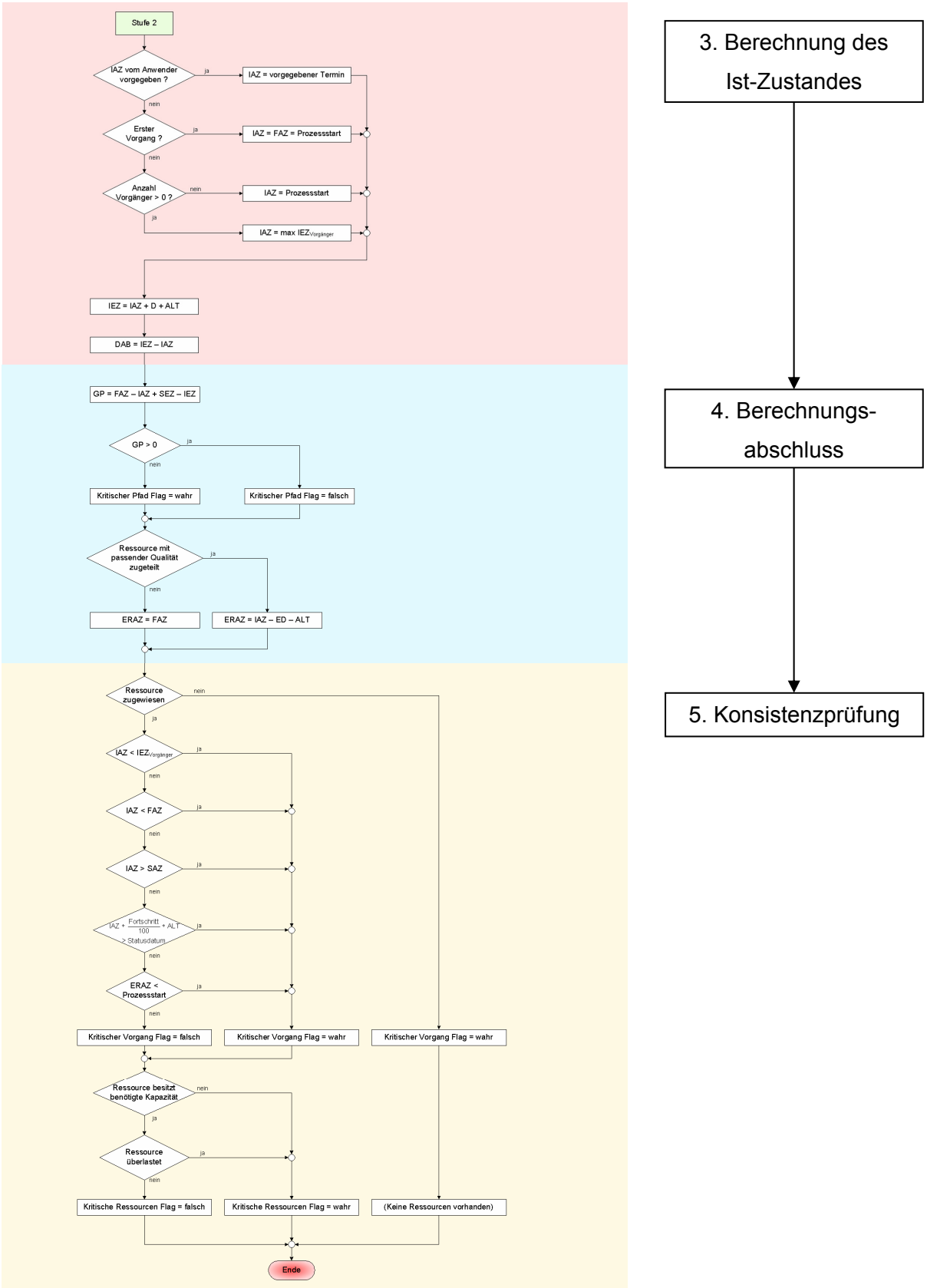


Abbildung 5-15: Ablauf des Berechnungsverfahrens zur Terminierung

Die einzelnen Stufen des Berechnungsverfahrens werden nachfolgend anhand des Fluss-diagramms dargelegt.

Eingangsgrößen

Die Terminierung ist von den in Abbildung 5-16 aufgeführten Eingangsgrößen abhängig. Nach jeder Änderung einer der Eingangsgrößen ist eine Neuberechnung der Terminierung erforderlich.

Der folgende Ausschnitt aus dem Flussdiagramm schlüsselt die Eingangsgrößen sortiert nach Zugehörigkeit auf. Pflichtangaben sind in dieser Ansicht schwarz hervorgehoben, während optionale Angaben grau dargestellt werden.

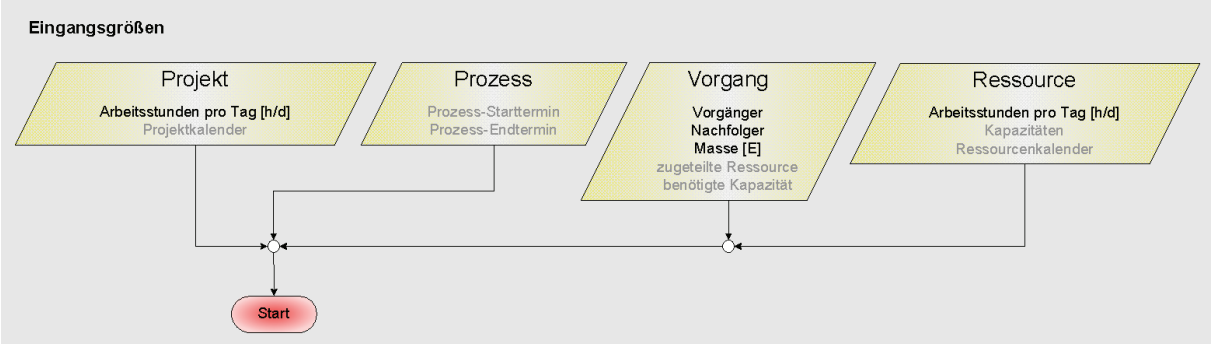


Abbildung 5-16: Ausschnitt Flussdiagramm, Eingangsgrößen

Die einzelnen Vorgänge eines konkreten Prozesses stehen im Mittelpunkt der Terminierung. Die Grundvoraussetzung für die nachfolgenden Berechnungen besteht darin, dass von jedem Vorgang die Masse sowie alle Vorgänger und Nachfolger bekannt sind.

Die Vorgänger und Nachfolger eines jeden Vorganges werden bei der Planung des Projektablaufes definiert. Dieser geht der Terminierung voraus und besteht aus der Definition der Anordnungsbeziehungen zwischen den Vorgängen des Prozesses. Dabei ist als Randbedingung zu beachten, dass als Vorgangsbeziehungen ausschließlich Normalfolgen[37] verwendet werden dürfen. Sollte die Verwendung von Normalfolgen aufgrund der Prozessstruktur nicht möglich sein, ist dies mittels Scheinvorgängen[38] zu kompensieren.

Weiterhin ist für die Terminierung eine quantitative Aussage über den Arbeitsaufwand der einzelnen Vorgänge erforderlich. Dieser wird durch Angabe der Masse erfasst. Die Masse kann entweder aus Erfahrungen abgeschätzt werden oder über Schätzungsmethoden ermittelt werden.

Neben den Angaben zu den Vorgängen werden als Eingangsgrößen zudem die Arbeitsstunden der Ressourcen benötigt. Weiterhin sind für die Terminierung die Standard-Arbeitsstunden des Projektes zwingend erforderlich.

---

[37] Normalfolge (Ende-Anfangs-Beziehung): Beziehung zwischen zwei Vorgängen, bei der das Ende des früheren Vorgangs Vorraussetzung für den Anfang des späteren Vorgangs ist

[38] Scheinvorgänge: fiktive Vorgänge der Dauer Null. Mit Hilfe von Scheinvorgängen können komplexe Anordnungsbeziehungen mittels begrenzter Vorgangsverknüpfungen abgebildet werden.

Zusätzlich zu den bisher aufgeführten Pflichtangaben können eine Reihe von optionalen Angaben getätigt werden. Die folgenden Stufen des Flussdiagramms verdeutlichen, dass wenn die optionalen Angaben nicht vorhanden sind, entweder Standardwerte in die Berechnung eingesetzt werden oder zusätzliche Berechnungen durchgeführt werden.

Stufe 1: Berechnung der Vorgangsdauer in Abhängigkeit von der Ressourcenkapazität

Die erste Stufe der Terminierung besteht in der Berechnung der Vorgangsdauer D. Im Gegensatz zu herkömmlichen Berechnungsverfahren erfolgt dies unter Berücksichtigung der Ressourcenkapazität. Das Flussdiagramms in Abbildung 5-17 verdeutlicht, dass in Abhängigkeit von den Eingangsgrößen die Vorgangsdauer auf drei verschiedene Arten bestimmt werden kann:

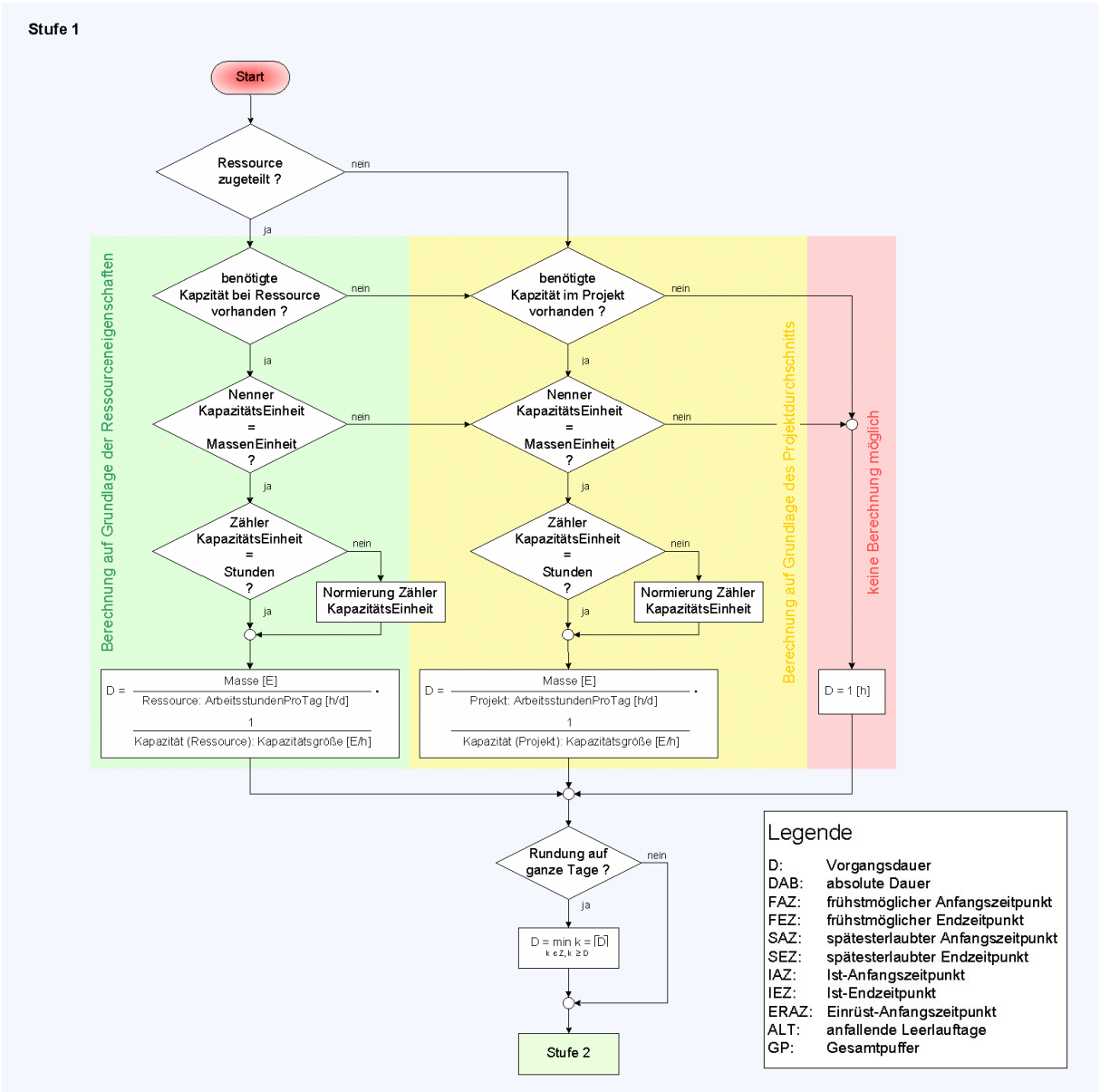


Abbildung 5-17: Ausschnitt Flussdiagramm, Vorgangsdauer

1. Wenn dem Vorgang eine Ressource zugewiesen ist, welche die kapazitiven Anforderungen des Vorganges erfüllt, wird die Dauer auf Grundlage der Eigenschaften der betreffenden Ressource berechnet. Gleichung 5-1 verdeutlicht, dass zur Berechnung der Vorgangsdauer die Masse durch die Arbeitsstunden und die Kapazitätsgröße der zugewiesenen Ressource dividiert wird.

$$D = \frac{\text{Masse [E]}}{\text{Ressource: ArbeitsstundenProTag [h/d]}} \cdot \frac{1}{\text{Kapazität (Ressource): Kapazitätsgröße [E/h]}}$$

Gleichung 5-1: Berechnung der Vorgangsdauer auf Grundlage der Ressourceneigenschaften

2. Wenn die Berechnung der Vorgangsdauer mittels der zugeteilten Ressource nicht möglich ist, wird sie anhand des Leistungsprofils des Projektes bestimmt.
   Diese Maßnahme ist notwendig, wenn einem Vorgang entweder keine Ressource zugeteilt wurde oder wenn die zugewiesene Ressource die kapazitiven Anforderungen des Vorganges nicht erfüllt. Dies ist der Fall, wenn die betreffende Ressource die vom Vorgang benötigte Kapazität nicht besitzt oder wenn die Einheit der Kapazität nicht mit der vorgegebenen Masse kompatibel ist.
   Unter diesen Voraussetzungen wird die Dauer des Vorganges aus seiner Masse sowie den Standard-Arbeitsstunden und der durchschnittlichen Größe der benötigten Kapazität errechnet. Diese Informationen stammen aus Kalender und Leistungsprofil des Projektes (vgl. Gleichung 5-2).

$$D = \frac{\text{Masse [E]}}{\text{Projekt: ArbeitsstundenProTag [h/d]}} \cdot \frac{1}{\text{Kapazität (Projekt): Kapazitätsgröße [E/h]}}$$

Gleichung 5-2: Berechnung der Vorgangsdauer auf Grundlage des Projektdurchschnitts

3. Für die Berechnung der Vorgangsdauer auf Grundlage von Durchschnittswerten ist es erforderlich, dass mindestens eine Ressource des Projektes die erforderliche Kapazität in der passenden Einheit besitzt.
   Wenn dies nicht der Fall ist, wird eine Dauer von einer Stunde angesetzt. Diese minimale Dauer ist notwendig, weil zum Abschluss der Terminierung alle Vorgänge eine positive Dauer besitzen müssen. In Berechnungsstufe 5 wird der Benutzer über ein Warn-Flag über diese Annahme informiert.

Die Einarbeitungs- bzw. Rüstzeit findet bei der Berechnung der Vorgangsdauern keine Berücksichtigung, weil sie in Berechnungsstufe 2 nicht eingeht. Die Einrüstung eines Vorganges kann bereits begonnen werden, bevor der Vorgängervorgang abgeschlossen ist. Dieser Sachverhalt wird in Abbildung 5-18 dargelegt. Der zweite Vorgang muss dort

aufgrund der Vorgangsbeziehungen direkt nach Beendigung des ersten Vorganges gestartet werden. Wann die Einrüstung von Vorgang 2 beginnt ist dafür irrelevant. Sie muss lediglich spätestens zeitgleich mit Vorgang 1 abgeschlossen werden, so dass Vorgang 2 ohne Verzögerung begonnen werden kann.

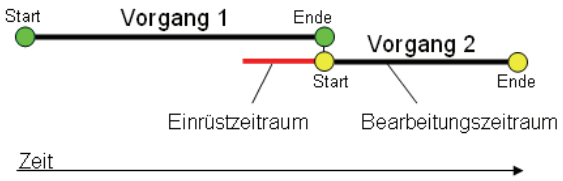


Abbildung 5-18: Visualisierung der Einrüstzeit

Weiterhin kann die Genauigkeit der Berechnung in Berechnungsstufe 1 gesteuert werden. Wenn die Projektplanung auf Grundlage von ganzen Tagen erfolgen soll, muss im Anschluss an diese Berechnungen die Dauer auf ganze Tage aufgerundet werden. Ansonsten wird die Terminierung mit Dezimalzahlen fortgesetzt.

Stufe 2: Berechnung der Vorgangsgrenzen durch Vorwärts- & Rückwärtsterminierung

In der zweiten Berechnungsstufe erfolgt die Berechnung der Vorgangsgrenzen. Dabei handelt es sich um den frühestmöglichen und den spätesterlaubten Zeitpunkt für Beginn und Abschluss eines Vorganges. Diese Termine definieren ein Zeitfenster, in denen ein Vorgang abgewickelt werden kann, ohne dass Prozessstart und Prozessende negativ beeinflusst werden oder Restriktionsverletzungen auftreten.
Für die Berechnung dieser Termine muss entweder der Start oder das Ende des zu dem Vorgang gehörigen Prozesses vorgegeben sein. Wenn dies nicht der Fall ist, wird das aktuelle Datum für den Prozessstart verwendet.

Die Berechnung der Vorgangsgrenzen untergliedert sich in zwei Teilschritte: die Vorwärts- und die Rückwärtsterminierung. Die Reihenfolge dieser Teilschritte hängt davon ab, von welchem Termin ausgehend die Berechnung erfolgt. Bei vorgegebenem Prozessstart wird von diesem Termin ausgehend zunächst eine Vorwärtsterminierung durchgeführt. Danach wird der Prozessendtermin berechnet und ausgehend von diesem Datum eine Rückwärtsterminierung abgewickelt. Wenn das Prozessende vorgeben ist, wird die Rückwärtsterminierung zuerst durchgeführt. Anschließend wird der Prozessstarttermin berechnet und ausgehend von diesem Datum eine Vorwärtsterminierung ausgeführt.

Abbildung 5-19 zeigt den Ausschnitt aus dem Flussdiagramm von diesem Teil des Berechnungsverfahrens.

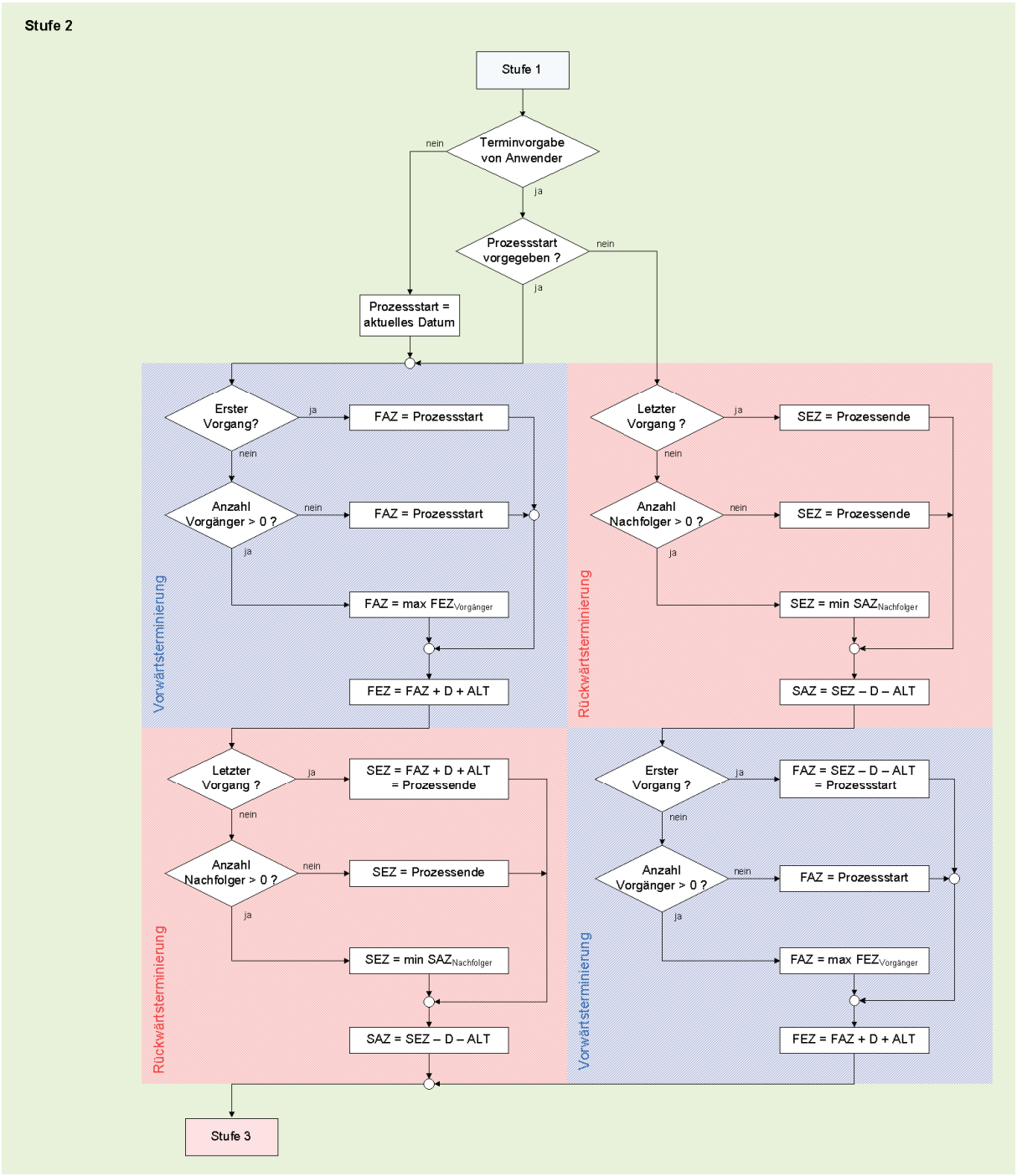


Abbildung 5-19: Ausschnitt Flussdiagramm, Vorgangsgrenzen

Bei der Vorwärtsterminierung werden ausgehend von dem Prozessstart mittels einer Vorwärtsrechnung der frühestmögliche Anfangszeitpunkt FAZ und der frühestmögliche Endzeitpunkt FEZ des Vorganges bestimmt. Der frühestmögliche Anfangszeitpunkt ergibt sich daraus, dass durch die Beschränkung auf Normalfolgen ein Vorgang erst dann begonnen werden kann, wenn alle seine Vorgänger abgeschlossen sind. Der frühestmögliche Anfangszeitpunkt ist somit gleich dem spätesten frühestmöglichen Endzeitpunkt FEZ aller Vorgänger ($FAZ = \max FEZ_{Vorgänger}$). Einzige Ausnahme bilden der erste Vorgang des Prozesses sowie

Vorgänge ohne Vorgänger. Bei ihnen wird der frühestmögliche Anfangszeitpunkt mit dem Prozessstart gleichgesetzt (FAZ = Prozessstart).
Der frühestmögliche Endzeitpunkt FEZ errechnet sich danach aus der Summe des frühestmöglichen Startzeitpunktes FAZ und der in Berechnungsstufe 1 ermittelten Vorgangsdauer D. Dabei ist allerdings zu berücksichtigen, dass sich der FEZ für jeden Leerlauftag der zwischen FAZ und FEZ liegt, um einen Tag in die Zukunft verschiebt. Daher wird an dieser Stelle die Größe ALT eingeführt, welche die Summe der anfallenden Leerlauftage beschreibt. Der frühestmögliche Endzeitpunkt ist somit FEZ = FAZ + D + ALT.

Abbildung 5-20 verdeutlicht den Ablauf der Vorwärtsterminierung anhand eines Prozesses welcher aus 4 Vorgängen besteht. Vorgang 1 ist Vorgänger von Vorgang 2 und Vorgang 3. Daher ist $FAZ_2 = FAZ_3 = FEZ_1$.

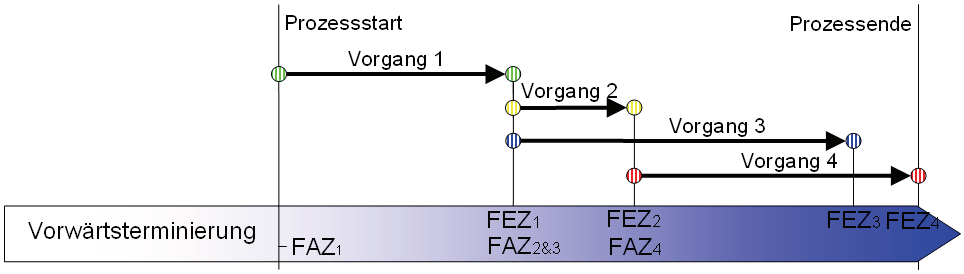


Abbildung 5-20: Vorwärtsterminierung

Bei der Rückwärtsterminierung werden ausgehend von dem Prozessende mittels Rückwärtsrechnung der spätestmögliche Endzeitpunkt SEZ sowie der frühestmögliche Endzeitpunkt SAZ aller Vorgänge bestimmt. Der spätestmögliche Endzeitpunkt ergibt sich daraus, dass durch die Beschränkung auf Normalfolgen ein Vorgang spätestens dann abgeschlossen sein muss, wenn der erste Nachfolgevorgang beginnt. SEZ entspricht daher dem frühesten spätestmöglichen Anfangszeitpunkt SAZ aller Nachfolger ($SEZ = \min SAZ_{Nachfolger}$). Ausnahmen bilden der letzte Vorgang des Prozesses sowie Vorgänge ohne Nachfolger. Bei ihnen wird SEZ mit dem Prozessende gleichgesetzt (SEZ = Prozessende).
Der spätesterlaubte Anfangszeitpunkt SAZ wird anschließend aus dem frühestmöglichen Endzeitpunkt errechnet indem die Vorgangsdauer D sowie die anfallenden Leerlauftage ALT abgezogen werden (SAZ = SEZ - D - ALT). Abbildung 5-21 stellt die Zusammenhänge bei der Rückwärtsterminierung grafisch dar. Vorgang 1 besitzt als Nachfolger die Vorgänge 2 und 3. Da der späteste Anfangszeitpunkt von Vorgang 2 vor dem von Vorgang 3 liegt, ist der späteste Endzeitpunkt von Vorgang 1 durch Vorgang 2 determiniert.

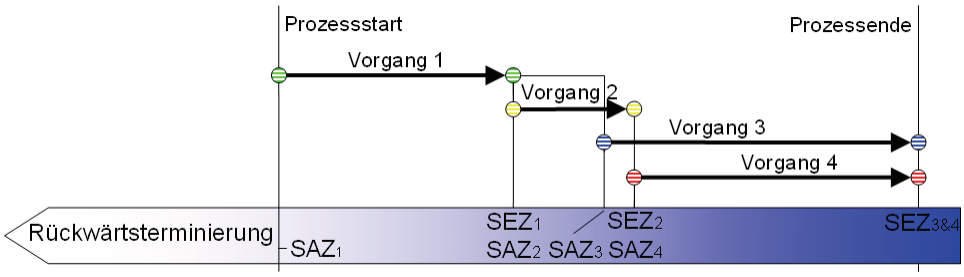


Abbildung 5-21: Rückwärtsterminierung

Stufe 3: Berechnung des Ist-Zustandes

Im Anschluss an die Vorwärts- und Rückwärtsterminierung erfolgt die Berechnung des Ist-Zustandes. Die zuvor ermittelten Vorgangsgrenzen definieren ein Zeitfenster, in denen ein Vorgang abgewickelt werden kann, ohne dass die Prozesstermine (Start, Ende) negativ beeinflusst werden oder Restriktionsverletzungen auftreten. Die tatsächliche zeitliche Lage eines Vorganges ist durch seinen Ist-Anfangszeitpunkt (IAZ) und Ist-Endzeitpunkt (IEZ) bestimmt. Abbildung 5-22 zeigt den Ausschnitt aus dem Flussdiagramm für die Bestimmung des Ist-Zustandes.

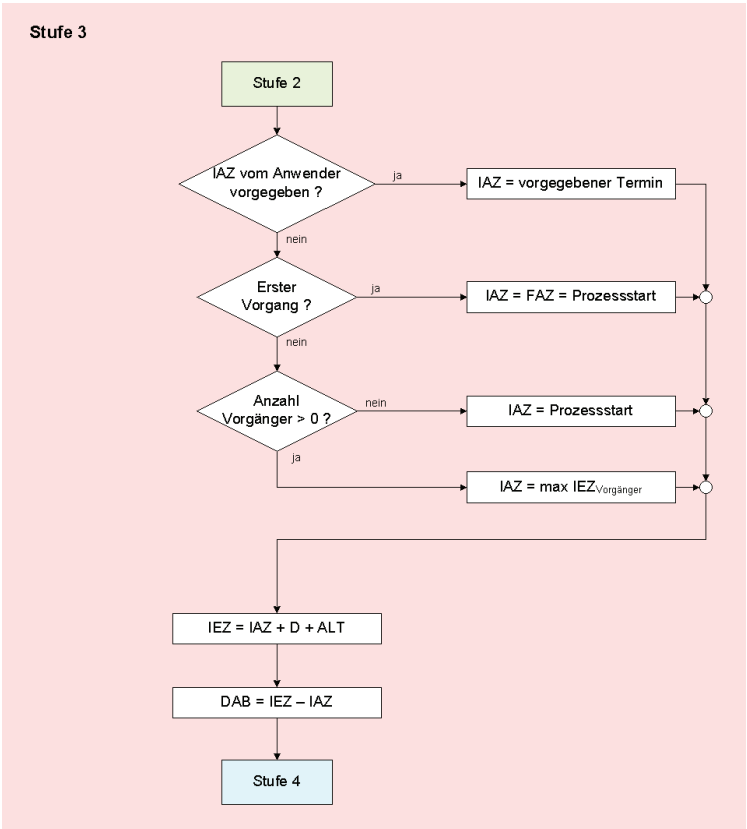


Abbildung 5-22: Ausschnitt Flussdiagramm, Ist-Zustand

Der Ist-Anfangszeitpunkt der einzelnen Vorgänge kann vom Anwender manuell vorgegeben werden (IAZ = vorgegebener Termin). Wenn dies nicht der Fall ist, wird der Vorgang automatisch in der frühestmöglichen Lage positioniert. Die dazu notwendigen Berechnungen gestalten sich analog zur Vorwärtsterminierung: Der Anfangszeitpunkt errechnet sich aus dem spätesten Endzeitpunkt aller Vorgänger (IAZ = max $IEZ_{Vorgänger}$), wobei der erste Vorgang des Prozesses sowie Vorgänge ohne Vorgänger Ausnahmen bilden. In diesen Fällen entspricht der Ist-Anfangszeitpunkt dem Prozessstart bzw. den frühestmöglichen Anfangszeitpunkt des Vorganges (IAZ = FAZ = Prozessstart). Der Ist-Endzeitpunkt IEZ errechnet sich aus dem Ist-Anfangszeitpunkt sowie der Vorgangsdauer und den in diesem Zeitraum anfallenden Leerlauftagen (IEZ = IAZ + D + ALT).
Weiterhin wird bei der Berechnung des Ist-Zustandes die absolute Vorgangsdauer DAB bestimmt. Diese ergibt sich aus der Anzahl von Tagen zwischen dem Ist-Anfangszeitpunkt und dem Ist-Endzeitpunkt (DAB = IEZ - IAZ).

Abbildung 5-23 verdeutlicht die Zusammenhänge zwischen den Werten der Ist-Terminierung. Während die Vorgänge 1,2 und 4 in der frühestmöglichen Lage positioniert sind (IAZ = FAZ), wurde Vorgang 3 innerhalb seiner Grenzen verschoben (FAZ <= IAZ <= SAZ und FEZ <= IEZ <= SEZ).

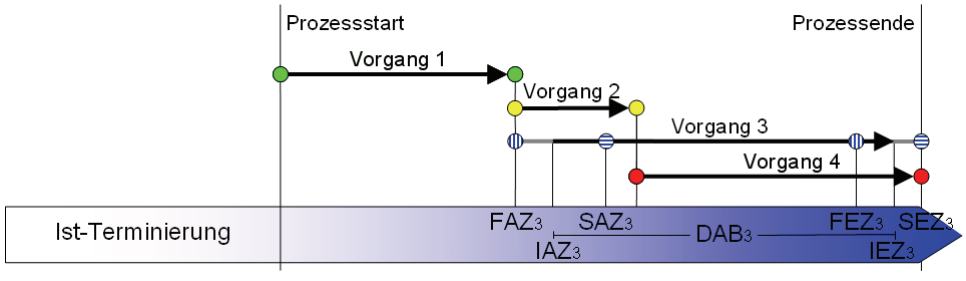


Abbildung 5-23: Ist-Terminierung

Stufe 4: Berechnungsabschluss

Der Abschluss der Berechnungsphase besteht in der Bestimmung der von den Vorgangsgrenzen und dem Ist-Zustand abhängigen Größen. Der zugehörige Ausschnitt aus dem Flussdiagramm ist in Abbildung 5-24 dargestellt.

Zunächst erfolgen die Berechnung des Gesamtpuffers und die Bestimmung des kritischen Pfades sowie die Terminierung der Einrüstzeit. Der Gesamtpuffer errechnet sich aus der Differenz zwischen dem Ist-Zustand und den Vorgangsgrenzen (GP = FAZ - IAZ + SEZ - IEZ). Laut Definition liegen alle Vorgänge, die keinen positiven Gesamtpuffer besitzen auf dem kritischen Pfad. Die Information, ob ein Vorgang auf dem kritischen Pfad liegt oder nicht, wird in Form eines Flags gespeichert, welches die Zustände wahr und falsch unterscheidet.

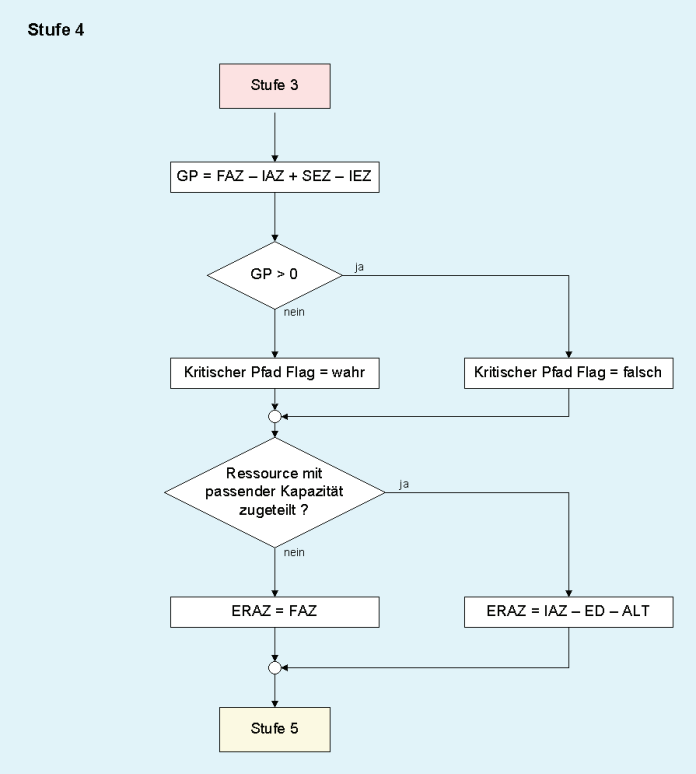


Abbildung 5-24: Ausschnitt Flussdiagramm, Berechnungsabschluss

Der Einrüst-Anfangszeitpunkt ERAZ wird ermittelt indem von dem Ist-Anfangszeitpunkt die Einrüstdauer ED sowie die anfallenden Leerlauftage subtrahiert werden (ERAZ = IAZ - ED - ALT). Da die Einrüstzeit kapazitätsabhängig ist, kann ein expliziter Einrüst-Anfangszeitpunkt nur für Vorgänge berechnet werden, die eine Ressource zugeteilt haben, welche die für den Vorgang benötigte Kapazität besitzt. Sollte dies nicht der Fall sein, wird der Einrüst-Anfangszeitpunkt mit dem frühestmöglichen Anfangszeitpunkt gleichgesetzt (ERAZ = FAZ) und im späteren Verlauf der Berechnung ein entsprechendes Warn-Flag aktiviert.

Die folgende Abbildung 5-25 illustriert den Zusammenhang zwischen dem Gesamtpuffer und dem kritischen Pfad. Die Vorgänge 1, 2 und 4 haben einen Gesamtpuffer von Null und liegen deshalb auf dem rot hervorgehobenen kritischen Pfad. Eine Verspätung eines dieser Vorgänge hätte eine sofortige Verlängerung des Prozesses zur Folge. Währenddessen besitzt Vorgang 3 einen positiven Gesamtpuffer, wodurch er nicht dem kritischen Pfad zugeordnet wird. Er kann innerhalb der Vorgangsgrenzen beliebig verschoben werden, ohne den Prozessstart- oder das Prozessende zu beeinflussen.
Weiterhin ist in Abbildung 5-25 Vorgang 4 mit einer Einrüstzeit belegt (blau), während bei den anderen Vorgängen eine vernachlässigbare Einrüstzeit vorliegt.

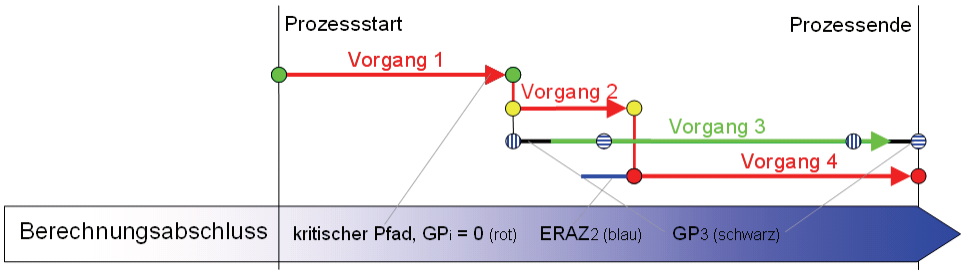


Abbildung 5-25: Berechnungsabschluss

Stufe 5: Konsistenzprüfung

Die bisher durchgeführten Berechnungen dienen der Bestimmung der zeitlichen Lage eines Vorganges. Sobald alle Vorgänge des Prozesses die Berechnungsstufen 1-4 durchlaufen haben, sind alle Informationen zur Erstellung eines Prozessplans vorhanden.

Da eine zentrale Anforderung an die P2-Plantafel in Benutzerfreundlichkeit besteht (vgl. Anforderungen in Kapitel 3), wird an die Berechnung der Vorgangsdaten eine Konsistenzprüfung angeschlossen. Diese untersucht die Benutzereingaben auf Vollständigkeit und Widerspruchsfreiheit. Die Ergebnisse der Konsistenzprüfung werden innerhalb entsprechender Flags gespeichert und ermöglichen in der Präsentationsschicht eine Hervorhebung inkonsistenter Vorgänge.

Als Beispiel sei an dieser Stelle eine in Berechnungsstufe 1 behandelte Thematik angeführt: Wenn einem Vorgang keine Ressource zugeordnet ist, erfolgt die Berechnung der Vorgangsdauer auf Grundlage der Durchschnittswerte des Projektes. Da die Vorgangsdauer direkten Einfluss auf sämtliche nachfolgenden Berechungsergebnisse ausübt, muss für die Projektleitung jederzeit ersichtlich sein, dass es sich bei den Daten des betreffenden Vorgangs um approximierte Werte handelt, die einer überschlägigen Planung dienen. Eine explizite Planung kann erst nach Zuweisung einer konkreten Ressource erfolgen.

Abbildung 5-26 illustriert den Abschluss des Berechnungsverfahrens durch die Konsistenzprüfung. Bei der Konsistenzprüfung werden sowohl der Vorgang als auch die zugeteilte Ressource analysiert. Einleitend wird geprüft, ob dem Vorgang überhaupt eine Ressource zugeordnet ist. Wenn dies nicht der Fall ist, wird der Vorgang als kritisch eingestuft (kritischer-Vorgang-Flag = wahr) und die Prüfung der zugeteilten Ressource übersprungen.

Wenn der Vorgang hingegen eine Ressource besitzt, wird eine Reihe von Bedingungen ausgewertet. Sobald eine der folgenden Bedingungen zutrifft, wird der Vorgang als kritischer Vorgang klassifiziert:

- Der Vorgang beginnt bevor alle Vorgänger beendet sind (IAZ < $IEZ_{Vorgänger}$).
- Der Vorgang beginnt vor dem frühestmöglichen Anfangszeitpunkt (IAZ < FAZ).
- Der Vorgang beginnt nach dem spätestmöglichen Anfangszeitpunkt (IAZ > SAZ).

• Der Vorgangsfortschritt liegt hinter der Planung zurück ($IAZ + \frac{Fortschritt}{100} + ALT > Statusdatum$).
• Die Einrüstung für den Vorgang beginnt vor dem Start des Prozesses.

Nach der Auswertung des Vorganges werden bei der zugeteilten Ressource folgende Bedingungen geprüft:

• Die Ressource besitzt eine zu der benötigten Kapazität des Vorganges passende Kapazität.
• Die Ressource ist überlastet.

Sollte die erste Bedingung nicht erfüllt werden oder die zweite Bedingung zutreffen, so ist die zugeteilte Ressource als kritisch einzustufen.

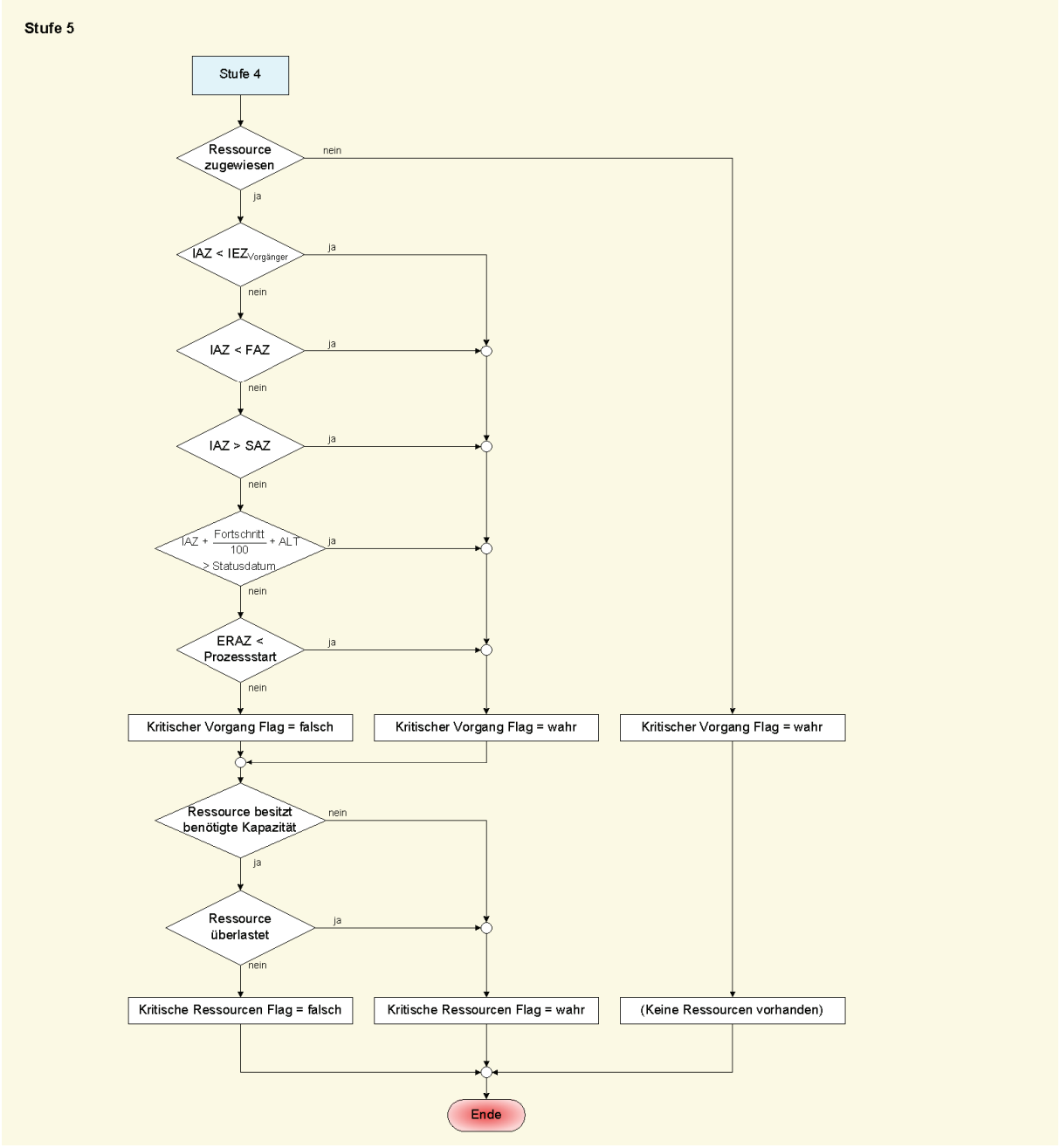


Abbildung 5-26: Ausschnitt Flussdiagramm, Konsistenzprüfung

5.5.3 Präsentationsschicht

Die Modellierung der Präsentationsschicht bildet den letzten Teilbereich der Konzeptionierung des Stand-Alone-Modus. Sie fungiert als Schnittstelle zwischen der zuvor thematisierten Logikschicht und dem Anwender.
Nachfolgend wird zunächst das grundlegende Konzept zur Datenrepräsentation umrissen, bevor die einzelnen Ansichten der Benutzeroberfläche detailliert werden. Im Anschluss daran erfolgt die Konkretisierung der zur Implementierung der Präsentationsschicht erforderlichen Klassen.

Abbildung 5-27 zeigt das grundlegende Konzept für die Benutzeroberfläche der P2-Plantafel. Sie stellt die Kerninformationen des Projektmanagements sowie die zugehörigen Daten aus dem PDM-System dar. Dazu werden fünf verschiedene Ansichten verwendet, die jeweils eine unterschiedliche Sicht auf die vorliegenden Daten bieten. Die ersten vier Ansichten fokussieren die drei elementaren Aspekte des Projektmanagements (Projekt, Vorgang und Ressource) sowie die Prozessdaten. Die fünfte Ansicht visualisiert Informationen aus dem PDM-System, die mit den in Ansicht 1-4 selektierten Objekten verknüpft sind. Da diese Komponente dem Integrationsmodul zuzuordnen ist, wird sie erst in dem zugehörigen Kapitel erläutert. Es folgt die Auswertung der Ansichten des Stand-Alone-Modus.

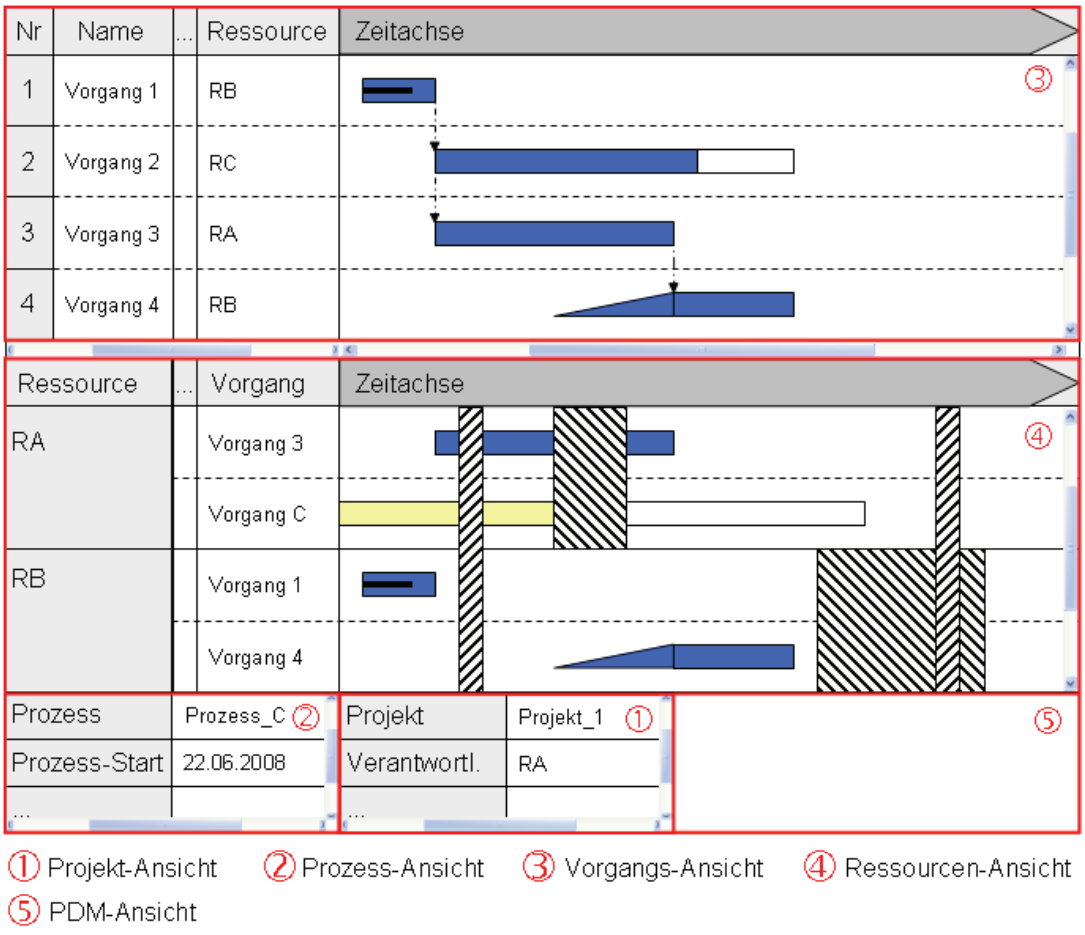


Abbildung 5-27: Plantafel-Oberfläche

Ansicht 1 (Projekt-Tabelle)

Das Projekt stellt die oberste Hierarchieebene der Projektplanung dar. Daher ist eine Ansicht, welche die grundlegenden Informationen zu dem in Bearbeitung befindlichen Projekt abbildet, zentraler Bestandteil der Benutzeroberfläche.

Ansicht 1 besteht aus einer Tabelle, welche alle Attribute des aktiven Projekt-Objektes zeigt. Da innerhalb der P2-Plantafel zwischen unterschiedlichen Projekten gewechselt werden kann, ist die wichtigste Information, die der Projekt-Tabelle entnommen werden kann, der Name des aktiven Projektes. Weiterhin werden der Projektverantwortliche, die Projektbeschreibung sowie die Termindaten für Projektstart, Projektende und die Statuszeit in Ansicht 1 abgebildet.

Da sich die Durchführung des Projektmanagements im Wesentlichen auf die Analyse von Vorgängen und Ressourcen konzentriert, werden die dazugehörigen Ansichten aus Gründen der Ergonomie oberhalb der Projekt-Tabelle angeordnet.

Ansicht 2 (Prozess-Tabelle)

Analog zur Projekt-Tabelle werden in der Prozess-Tabelle die grundlegenden Informationen zu dem in Bearbeitung befindlichen Prozess angezeigt. Da diese Ansicht häufiger benötigt wird als die Projekttabelle, wird sie auf der linken Seite angeordnet, während die Projekttabelle rechts davon positioniert wird. Die Anordnung der Ansichten unterstützt somit das europäische Leseverhalten (von links nach rechts).

Ansicht 3 (Gantt-Diagramm)

Ansicht 3 dient der Darstellung der Struktur des aktiven Prozesses. Zudem werden alle Informationen zu den einzelnen Vorgängen des betreffenden Prozesses in dieser Ansicht abgebildet.

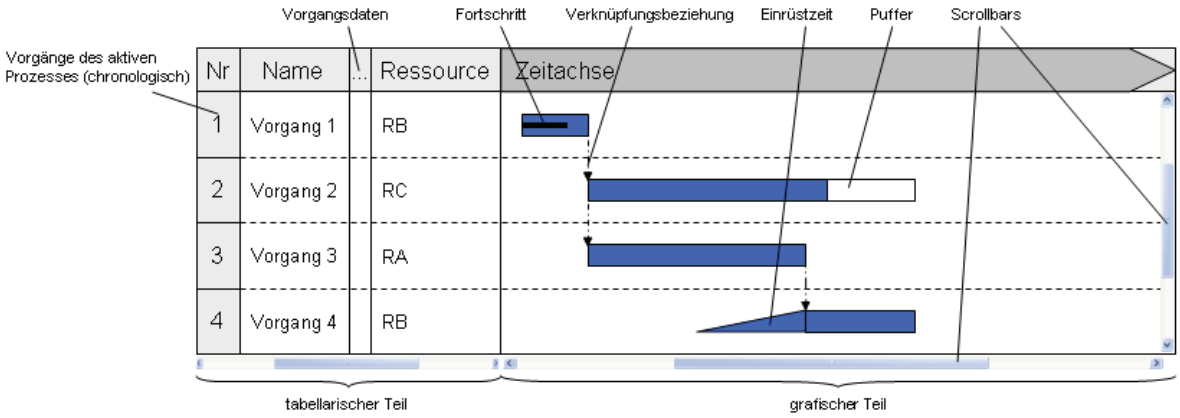


Abbildung 5-28: Ansicht 3, Gantt-Diagramm

Der Aufbau von Ansicht 3 beruht auf einem erweiterten Gantt-Diagramm. Ein Gantt-Diagramm besteht aus einem tabellarischen und einem grafischen Teil (vgl. Abbildung 5-28). In der Tabelle sind Informationen zu den Prozessvorgängen in Textform hinterlegt, während in der Grafik dieselben Informationen in Form von übereinander angeordneten Balken wiedergegeben werden. Der zu einem Vorgang gehörige Balken befindet sich dabei immer auf derselben Höhe wie die dazugehörige Tabellenzeile, während seine horizontale Lage durch eine Zeitachse bestimmt ist. Änderungen können entweder über die Tabelle eingegeben werden (dann wird der Balken entsprechend angeglichen) oder mittels Drag-&-Drop[39] am Balken vorgenommen werden (dann wird die Tabelle aktualisiert).
Um die Abhängigkeiten zwischen den Vorgängen abzubilden, werden im Gantt-Diagramm Verknüpfungsbeziehungen angezeigt. Diese werden in der Grafik in Form von Pfeilen visualisiert. Diese Weiterentwicklung des ursprünglichen Gantt-Diagramms wird als vernetztes Gantt-Diagramm, vernetzter Balkenplan oder Plannet-Diagramm bezeichnet.

Im grafischen Teil des Gantt-Diagramms werden für die Anzeige von Planungsinformationen unterschiedliche Elemente verwendet, wobei die Breite aller Elemente proportional zu deren Dauer ist. Aus der Lage der farbigen Balken sind die Dauer eines Vorganges sowie sein Start- und Endtermin ersichtlich. Pufferzeiten werden durch transparente Balken angezeigt, die an den farbigen Balken angrenzen. Einrüstzeiten werden durch ein ansteigendes Balkensegment dargestellt. Außerdem wird der Arbeitsfortschritt der einzelnen Vorgänge durch ein schwarzes Balkensegment hervorgehoben, dessen Länge proportional zu dem Arbeitsfortschritt in Prozent ist.

Zusätzlich zu dem Gantt-Diagramm des aktiven Prozesses können Gantt-Diagramme von anderen Prozessen angezeigt werden. Dadurch wird ein Vergleich von verschiedenen Prozessen ermöglicht. Zudem können die einzelnen Vorgänge einer Ressource nur im Kontext des dazugehörigen Prozesses beurteilt werden. Für das Management von Ressourcen, die Vorgängen aus verschiedenen Prozessen bearbeiten, ist diese Funktion somit unerlässlich (vgl. Ressourcendiagramm).
Beim Öffnen von zusätzlichen Gantt-Diagrammen wird durch entsprechende Zugriffsmechanismen sichergestellt, dass vom Anwender nur Prozesse bearbeitet werden können, auf die er Schreibzugriff besitzt. Die restlichen Prozesse können lediglich im Lesemodus betrachtet werden.

[39] Drag-&-Drop: <engl. Ziehen und Fallenlassen> Methode zum Bewegen grafischer Elemente auf einer Benutzeroberfläche mit Hilfe einer Maus

Ansicht 4 (Ressourcen-Diagramm)

Ansicht 4 dient der Planung des Ressourceneinsatzes. Wie bereits im Rahmen der Terminierung ausgeführt wurde, übt die Ressourcenzuteilung direkten Einfluss auf die Dauer der Vorgänge aus. Die Berücksichtigung von Ressourcenverfügbarkeit und –auslastung bildet daher einen wesentlichen Aspekt des Projektmanagements. Aus diesem Grund ist es erforderlich, dass diese Informationen ständig verfügbar sind. Daher werden die ressourcenspezifischen Daten bei der P2-Plantafel in einer zentralen Ansicht dargestellt.

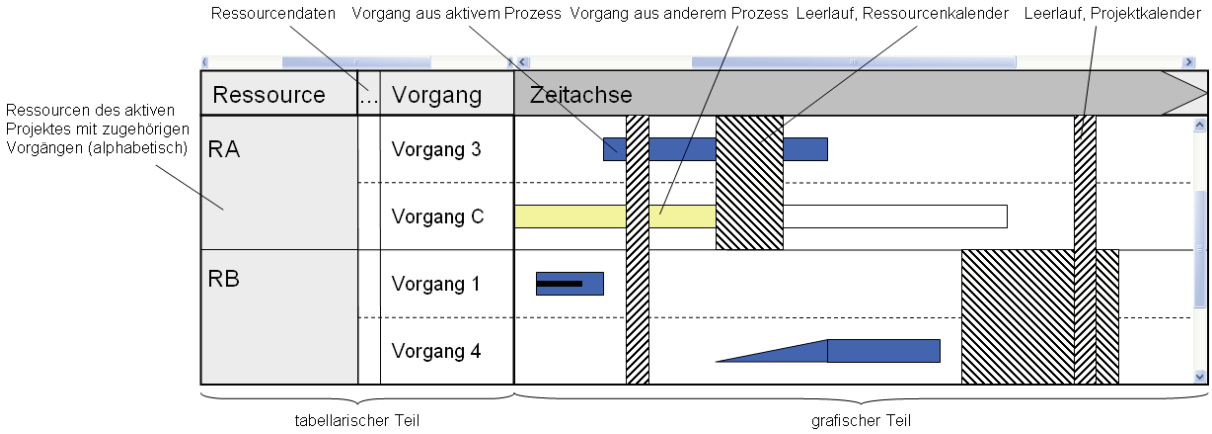


Abbildung 5-29: Ansicht 2: Ressourcen-Diagramm

Um eine durchgehende Benutzerführung zu gewährleisten erfolgt der Aufbau des Diagramms zur Abbildung der Ressourcendaten analog zu dem des Gantt-Diagramms. Allerdings werden in dem Ressourcen-Diagramm nicht die Vorgänge in chronologischer Reihenfolge angezeigt, sondern die Ressourcen in alphabetischer Reihenfolge (siehe Abbildung 5-29).

In dem tabellarischen Teil des Ressourcen-Diagramms werden für alle Ressourcen aus dem Pool des aktiven Projektes die zugeteilten Vorgänge angegeben und die zugehörigen ressourcenspezifischen Daten aufgeschlüsselt. Dies umfasst Kapazität, Produktivitätsfaktor und Einarbeitungszeit.
Für die Auswertung der Ressourcenauslastung ist es erforderlich, dass sämtliche Arbeitsaufgaben aller Ressourcen erfasst werden. Daher ist das Ressourcendiagramm nicht auf den aktiven Prozess beschränkt, sondern es werden sämtliche Vorgänge abgebildet, an denen die Ressourcen des Projektes beteiligt sind.

Der grafische Teil des Ressourcen-Diagramms visualisiert die Vorgänge aller Ressourcen des aktiven Projektes. Vorgänge, die nicht zu dem in Bearbeitung befindlichen Prozess gehören, werden dabei durch eine andere Farbe hervorgehoben. Da die Sortierung der Vorgänge nach Ressourcen geordnet erfolgt, werden die Vorgangsbeziehungen in dem

Diagramm nicht angezeigt. Stattdessen werden als Zusatzinformation, die Leerlaufzeiten der Ressourcen, abgebildet. Dabei erfolgt die Unterscheidung von Leerlaufzeiten aus dem Projektkalender und Leerlaufzeiten aus dem Ressourcenkalender mittels einer gegenläufigen Schraffur.

Die aus Projekt- und Prozess-Tabelle sowie Vorgangs- und Ressourcendiagramm bestehende Benutzeroberfläche bietet einen transparenten Überblick über ein komplettes Entwicklungsprojekt und alle referenzierten Objekte. Somit wird das zentrale Defizit von den derzeit verfügbaren PM-Anwendungen, dass lediglich einzelne Prozesse eines Projektes abgebildet werden können, von dem entwickelten Konzept komplett behoben.

Neben den vorgestellten Hauptansichten können zudem über Dialogfelder zusätzliche Informationen eingesehen werden. Dabei handelt es sich beispielsweise um den kompletten Kalender einer Ressource oder eine Ansicht, welche die zeitliche Lage der einzelnen Prozesse eines Projektes visualisiert.
Weiterhin dient die Plantafeloberfläche als Benutzerschnittstelle. Dies umfasst neben der reinen Repräsentation der Daten auch die Interaktion mit dem Anwender. Daher sind in die Benutzeroberfläche entsprechende Steuerelemente und Menüs eingearbeitet.

Die zur Implementierung der Präsentationsschicht erforderlichen Klassen werden in dem nachfolgenden Klassendiagramm konkretisiert (siehe Abbildung 5-30). Es zeigt einen Auszug aus den Klassendiagramm des kompletten PM-Moduls (vgl. Abbildung 5-11). Der Fokus dieses Diagramms liegt auf den zuvor erläuterten Ansichtsklassen. Die Klassen der Logikschicht sowie Klassen zur Benutzerinteraktion werden aus Gründen der Übersichtlichkeit nicht explizit aufgeführt. Die Thematik der Menüs, Dialogfelder und Steuerelemente wird in Kapitel 6 im Rahmen der Ausarbeitung des Anwendungsprototypen vertieft.

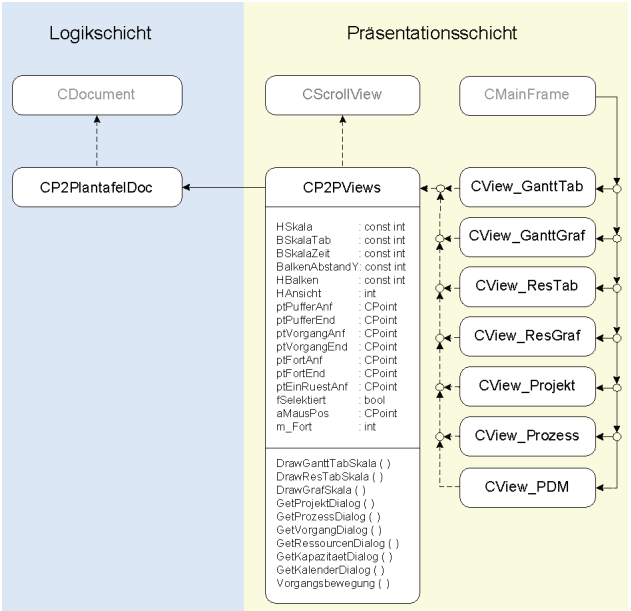


Abbildung 5-30: Klassendiagramm des PM-Moduls (Auszug)

Da die P2-Plantafel auf der Dokumenten-Ansicht-Architektur beruht, werden eine Vielzahl der für die Datenpräsentation und Benutzerinteraktion erforderlichen Klassen und Methoden vom MFC-Framework bereitgestellt. Für die Präsentationsschicht der Plantafel ist daher primär die Konzeptionierung der Ansichtsklassen erforderlich. Der Datenaustausch der Ansichtsklassen mit der Dokumentklasse **CP2PlantafelDoc**, die Erstellung der Dokumentvorlage sowie die Einbettung der Ansichten in ein Rahmenfenster, welches die Menü-, Symbol- und Statusleisten bereitstellt, ist Teil der MFC-Funktionalität und wird daher an dieser Stelle nicht behandelt. Nähere Informationen sind aus der zugehörigen Fachliteratur zu entnehmen.

Die Mehrzahl der Ansichtsklassen müssen nur in geringem Umfang eigene Eigenschaften implementieren. So ist beispielsweise die Struktur des Gantt-Diagramms und des Ressourcen-Diagramms identisch. Sie unterscheiden sich lediglich in der Anordnung der Vorgänge und durch die in der Tabelle angezeigten Daten. Weiterhin werden in allen Ansichten für dieselben Elementtypen die gleichen Kontextmenüs und Dialogfelder angezeigt.
Aus diesen Gründen werden alle wesentlichen Attribute und Methoden der Ansichtsklassen in einer zentralen Klasse implementiert und an alle Ansichtsklassen vererbt. Die Klasse ***CP2PViews*** fungiert als abstrakte Basisklasse[40] für die Ansichtklassen der P2P. Sie ist von der MFC-Klasse ***CScrollView*** abgeleitet, die als Basisklasse für Ansichten mit vordefinierten Scrollbars dient. Zusätzlich zu den aus ***CScrollView*** stammenden Ansichtsfunktionen beinhaltet ***CP2PViews*** allgemeine Daten, wie Koordinatenwerte und Farben sowie Methoden zur grafischen Ausgabe. In den P2P-Ansichten müssen daher lediglich ansichtsspezifische Spezifikationen ausgearbeitet werden. Im Falle der Tabelle des Gantt-Diagramms (***CView-GanttTab***) ist beispielsweise eine Definition erforderlich welche Daten in welcher Spalte eingetragen werden.

Auf ein Objektdiagramm der Präsentationsschicht wird an dieser Stelle verzichtet. Zur Laufzeit existiert von jeder Ansichtsklasse immer genau ein Objekt, während von der abstakten Basisklasse ***CP2PViews*** kein Objekt instanziert wird.

[40] Abstrakte Klasse: Klasse von der keine Objekte instanziiert werden können.

5.6 Integrationsmodus

In Kapitel 5.5 erfolgte die Konzeption des Stand-Alone-Modus der P2-Plantafel. Diese Ausbaustufe der Plantafel bildet eine eigenständige Anwendung, die vollständig an die Anforderungen der EDV-Unterstützung von Entwicklungsprojekten angepasst ist.
In der zweiten Entwicklungsstufe erfolgt die Detaillierung des Konzeptes zum Datenaustausch zwischen der Plantafel-Anwendung und PDM-Systemen. Das Ziel dieser Technologie besteht in der Erschließung zusätzlicher Optimierungspotenziale, durch Realisierung einer durchgängigen EDV-Unterstützung für die Produktentwicklung. Zu diesem Zweck muss die P2-Plantafel Planungsergebnisse an ein konkretes PDM-Systeme übertragen können und uneingeschränkten Zugriff auf die Daten und Dokumente dieses PDM-Systems bieten.
Da in der zweiten Entwicklungsstufe im Gegensatz zu dem Stand-Alone-Modus eine Integration der P2-Plantafel in ein PDM-System realisiert wird, trägt diese Technologie die Bezeichnung Integrationsmodus.

Im Integrationsmodus kann von der Projektleitung die komplette Planung und Steuerung der im PDM-System vorliegenden Projekte über die Plantafel ausgeführt werden. Die Durchführung der täglichen Arbeitsabläufe wird währenddessen weiterhin von den Anwendern des PDM-System durchgeführt. Dies erfolgt allerdings unter den innerhalb der P2P festgelegten Randbedingungen. Die innerhalb der Plantafel determinierten Termine dienen beispielsweise als Vorgabe für Beginn und Ende der im PDM-System abgebildeten Vorgänge.
Abgesehen davon unterstützt die Plantafel die Projektteams bei der Projektabwicklung indem sie zu jeder Zeit einen transparenten und aktuellen Überblick über sämtliche Aspekte des Projektes bereitstellt und direkten Zugriff auf alle relevanten Dokumente ermöglicht. Durch entsprechende Sicherheitsmechanismen wird dabei gewährleistet, dass Änderungen an der Projektplanung lediglich von der Projektleitung durchgeführt werden können. Den restlichen Projektmitgliedern bleibt ein lesender Zugriff vorbehalten.
Die Plantafel dient folglich nicht ausschließlich als operatives System der Projektleitung, sondern auch als Informationssystem für alle Projektbeteiligten. Dies stellt neben der optimierten Planung und Steuerung von Projekten einen weiteren zentralen Nutzen des entwickelten Konzeptes dar.

Abbildung 5-31 verdeutlicht die Architektur des Gesamtkonzeptes der P2-Plantafel, welches sich aus dem Stand-Alone- und dem Integrationsmodus zusammensetzt. Es folgt die Konzeptionierung des Integrationsmodus.

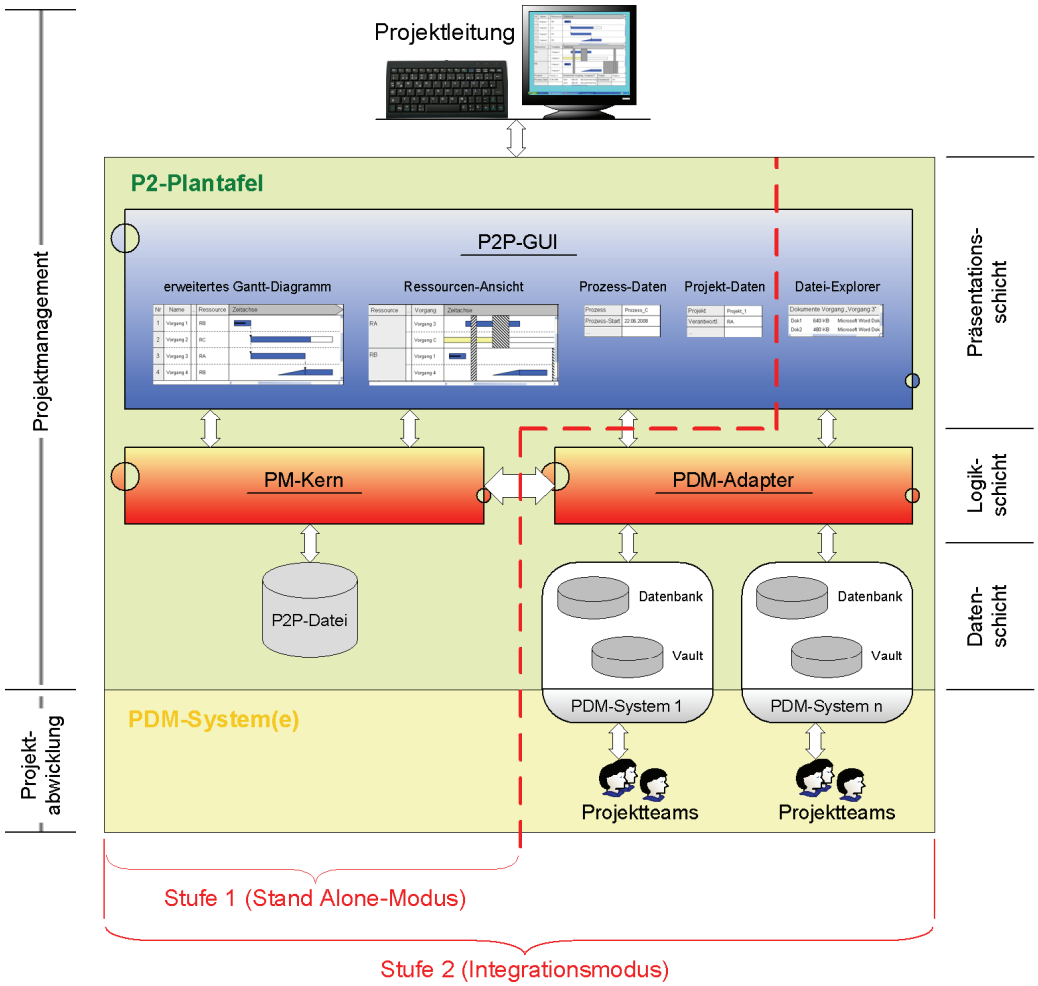


Abbildung 5-31: Entwicklungsstufen der P2P

Die Konzeption des Integrationsmodus wird anhand der Integration der P2P mit einem konkreten PDM-System dargelegt. Aufgrund der langjährigen Erfahrung am Institut für Ingenieurinformatik im Umfeld der Inbetriebnahme und Erweiterung von PDM-Systemen wurde für die Integration das PDM-System SmarTeam der Firma Dassault Systems ausgewählt. Dassault Systems ist einer der Marktführer im Bereich CAD, CAM und CAE und vertreibt SmarTeam mit großem internationalem Erfolg als PDM-Lösung für kleine und mittlere Unternehmen.

SmarTeam erfüllt die im Kapitel 3 gestellten Anforderungen an ein PDM-System vollständig. Durch seine offene Systemarchitektur und eine außergewöhnlich hohe Anpassbarkeit ist SmarTeam für die Konzeption der P2-Plantafel besonders gut geeignet.

Derweil wird durch die hohe Flexibilität des P2P-Konzeptes gewährleistet, dass unter geringem Aufwand jederzeit ein beliebiges, anderes PDM-System in die Anwendungsarchitektur der P2-Plantafel integriert werden kann.

Bei der Konzeptionierung des Integrationsmodus wird analog zum Stand-Alone-Modus von Innen nach Außen vorgegangen.

5.6.1 Datenschicht

Im Integrationsmodus wird zwischen der P2-Plantafel und dem PDM-System ein bidirektionaler Datenaustausch etabliert. Für die Synchronisation der Daten ist ein kompatibles Datenmodell eine zwingende Voraussetzung. Aus diesem Grund ist es erforderlich, dass das Datenmodell des PDM-Systems an das Datenkonzept der Plantafel angepasst wird. Weiterhin muss das Datenmodell der P2P dahingehend erweitert werden, so dass es von der Datenbank des PDM-Systems verwaltet werden kann.

Die Konzeption des Stand-Alone-Modus erfolgte auf Grundlage eines einleitenden Vergleichs zwischen PDM- und PM-Systemen (vgl. Kap. 5.2). Es wurde aufgezeigt, dass nahezu alle zentralen Objekte der Plantafel PDM-konform sind. Daher ist das grundlegende Datenmodell, welches für das Datenkonzept der Plantafel erforderlich ist, innerhalb des PDM-Systems bereits vorhanden.
Weiterhin handelt es sich bei der P2-Plantafel um eine Anwendung, deren Architektur für die Integration mit verschiedenen PDM-Systemen konzipiert wurde. Aus diesen Gründen liegt auf der Ebene der Datenschicht sowohl Plantafel- als auch PDM-seitig ein geringer Integrationsaufwand vor.
Nachfolgend werden die für eine Integration erforderlichen Modifikationen an dem Datenmodell der P2-Plantafel sowie an dem PDM-Datenmodell diskutiert.

Anpassung des Datenmodells der P2-Plantafel

Damit die Objekte der Plantafel von der Datenbank des PDM-Systems verwaltet werden können, muss eine eindeutige Identifizierung sämtlicher synchronisierbarer Objekte vorhanden sein. Daher ist es erforderlich, dass die Klassen ***CProjekt***, ***CProzess***, ***CVorgang***, ***CRessource***, ***CKapazitaet*** und ***CLeerlauf*** des Stand-Alone-Modus um die Attribute ***Class-ID (CID)*** und ***Object-ID (OID)*** erweitert werden. Diese Attribute vom Datentyp int bzw. long werden in der P2P mit dem Wert 0 initialisiert und beim Speichern in SmarTeam mit einer Identifikationskennung belegt. Diese Kennung bleibt über den kompletten Lebenszyklus eines Objektes unverändert und wird bei jedem Im- und Export eines Objektes übertragen.

Anpassung des PDM-Datenmodells

PDM-seitig ist für die Etablierung des Integrationsmodus eine Erweiterung des SmarTeam-Datenmodells notwendig. Im Einzelnen sind dabei die Klassen zur Abbildung von Projekten, Prozessen und Vorgängen um die in Kapitel 5.5 definierten PM-spezifische Attribute zu erweitern. Weiterhin sind die wenigen in SmarTeam nicht vorhandenen Klassen einzufügen (***CKalender*** und ***CKapazitaet***) sowie die PM-spezifischen Verknüpfungen zwischen den Klassen zu definieren. Die Detaillierung dieser Komponenten erfolgte ebenfalls in Kapitel 5.5.

Das PDM-Systems SmarTeam verwendet für die Verwaltung aller Informationen ein objektorientiertes Datenmodell, welches auf einer relationalen Datenbank aufsetzt.
Sämtliche Informationen werden innerhalb von SmarTeam in Form von Objekten repräsentiert, die spezifische Attribute und Methoden besitzen. Auch die Verknüpfungen zwischen verschiedenen Objekten sowie zwischen Objekten und Dokumenten werden mittels entsprechender Objekte erfasst.
Die Verwaltung der Informationen der einzelnen Objekte erfolgt in einer relationalen Datenbank. Mittels komplexer Transformationen werden die Informationen der Objekte in den Tabellen der Datenbank abgebildet.
Die relationale Ebene der Datenverwaltung wird jedoch von SmarTeam mittels entsprechender Hilfsprogramme komplett verborgen, so dass sich dem Anwender das System als lückenlos objektorientiert präsentiert. Die Definition der einzelnen Klassen wird dabei mit Hilfe des SmarTeam-Hilfsprogramms "Data Modell Designer" durchgeführt. Dieses Hilfsprogramm ermöglicht es, das Datenmodell des PDM-Systems objektorientiert zu bearbeiten. Die relationale Datenbank des Systems wird dabei im Hintergrund automatisch aktualisiert. Die guten Performanceeigenschaften von relationalen Datenbanken werden so mit den Prinzipien der objektorientierten Programmierung kombiniert.

Um die Spezifikationen der P2-Plantafel in das SmarTeam-Datenmodell zu integrieren sind im Einzelnen folgende Schritte erforderlich:

1. Klassendefinition
2. Definition der Attribute der Klassen
3. Definition der Beziehungen zwischen den Klassen
4. Definition der Profilkarten

1. Klassendefinition

Der erste Schritt bei der Überarbeitung des SmarTeam-Datenmodells besteht in der Definition der für die P2P-Integration erforderlichen Klassen bzw. in der Änderung der im Datenmodell bereits vorhandenen Klassen. Bei der Klassendefinition werden neben den Namen der Klasse auch die grundlegenden Eigenschaften der Klassen festgelegt. Diese Eigenschaften steuern welche Mechanismen auf die erzeugten Objekte angewandt werden. Eine charakteristische Klasseneigenschaft ist beispielsweise die "Versionssteuerung". Mittels dieser Eigenschaft wird definiert, ob die Objekte der Klasse an den LifeCyle-Operationen teilnehmen sollen und versioniert werden können.
Abgesehen davon werden bei der Klassendefinition mittels Anordnung der Klassen innerhalb einer Baumstruktur die Vererbungsbeziehungen zwischen den einzelnen Klassen festgelegt (vgl. Abbildung 5-32). [33]

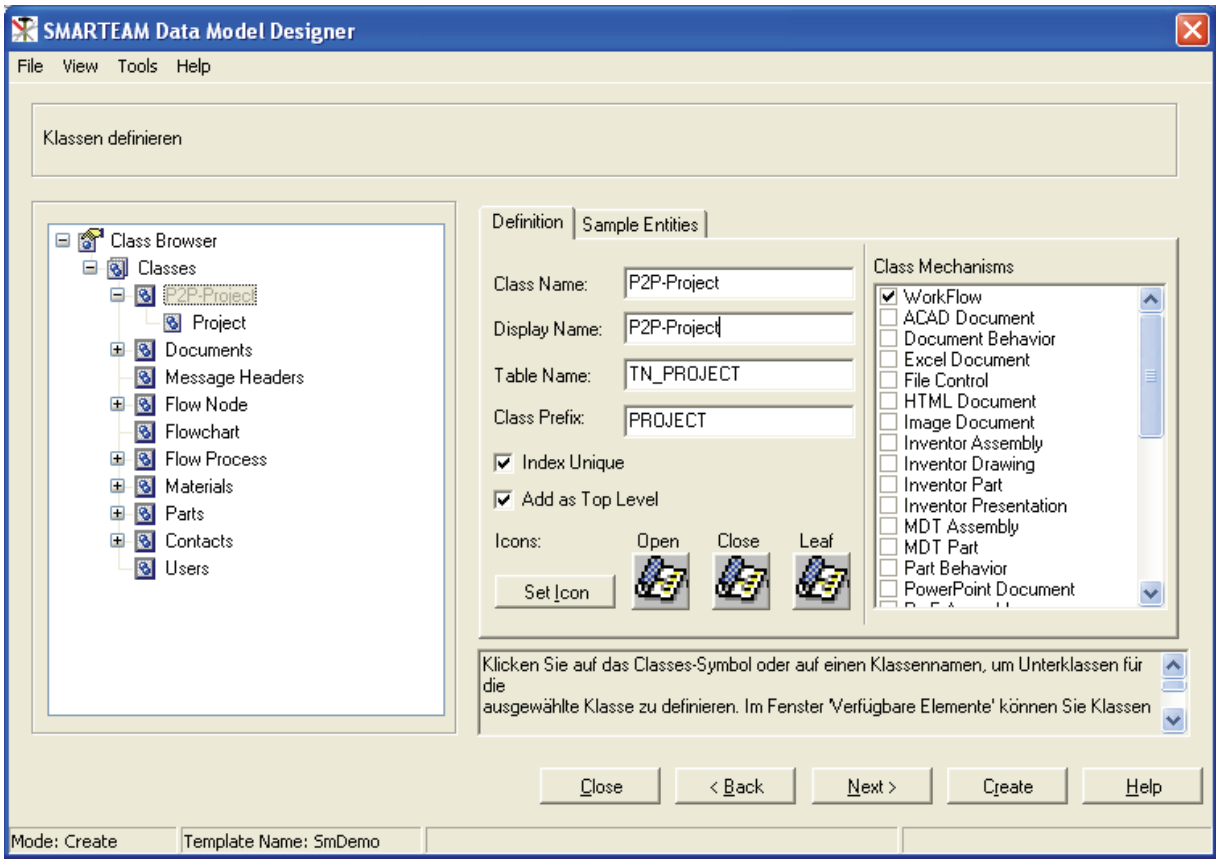


Abbildung 5-32: Klassendefinition mit dem SmarTeam Data Model Designer

2. Definition der Attribute der Klassen

Im Anschluss an die Klassendefinition sind die Attribute der einzelnen Klassen zu konkretisieren. Die Attribute aller SmarTeam-Klassen können über die aus der Klassendefinition bekannte Baumstruktur eingesehen und geändert werden (vgl. Abbildung 5-33). Dabei bietet SmarTeam die Möglichkeit, eine Vielzahl unterschiedlicher Datentypen zu verwenden. Die Auswahl der verfügbaren Datentypen reicht von in der Programmierung gängigen Typen, wie Zahlen oder Zeichenketten, bis hin zu systemspezifischen Datentypen, wie Zeitstempel oder "Bezug zu einem anderen Objekt".

3. Definition der Beziehungen zwischen den Klassen

Wie im Rahmen der Konzeption des Stand-Alone-Modus bereits ausgeführt wurde, sind für die vollständige Abbildung sämtlicher Informationen des Projektmanagements vielfältige Beziehungen erforderlich. Der dritte Schritt der Klassenmodellierung mit dem Data Model Designer umfasst die Definition der Beziehungen zwischen den Objekten des PDM-Systems (siehe Abbildung 5-34). Dabei werden sowohl einfache Referenzen als auch logische Links unterstützt.

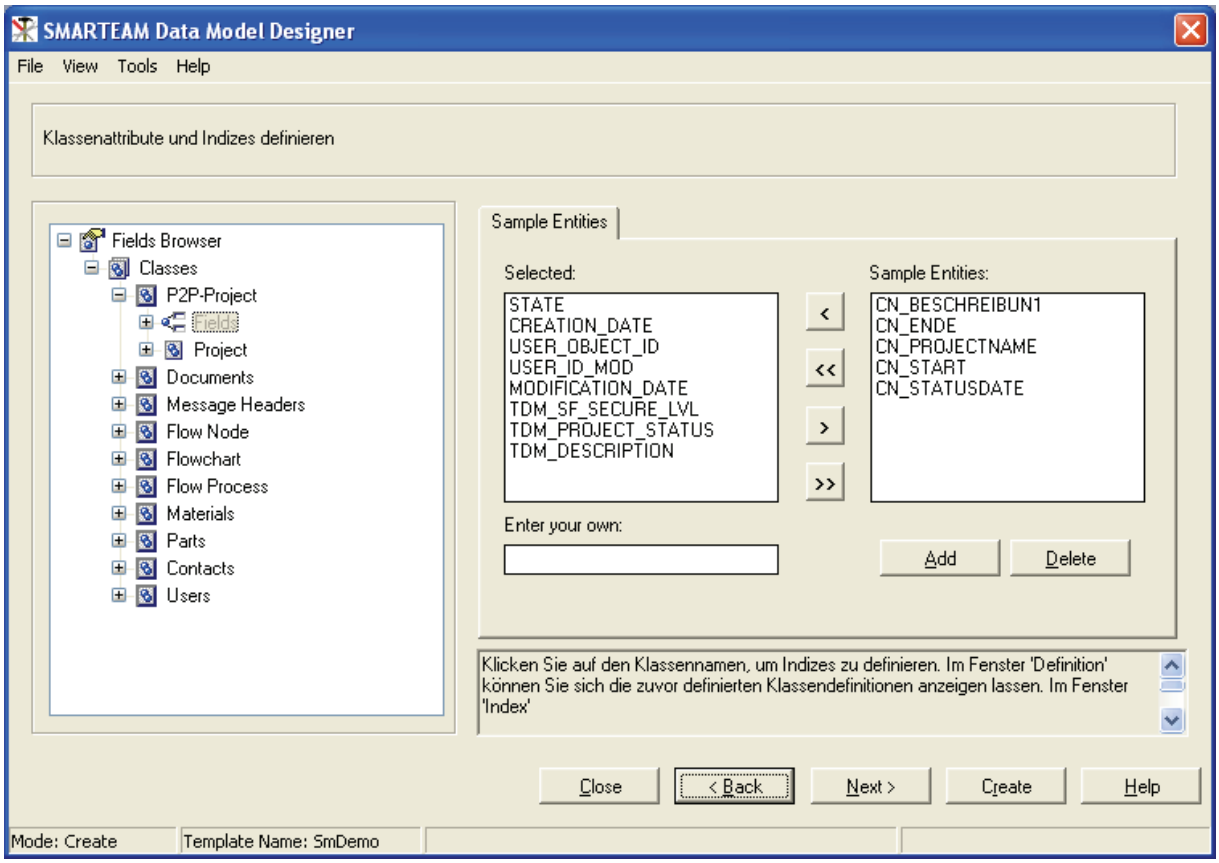


Abbildung 5-33: Attributedefinition mit dem SmarTeam Data Model Designer

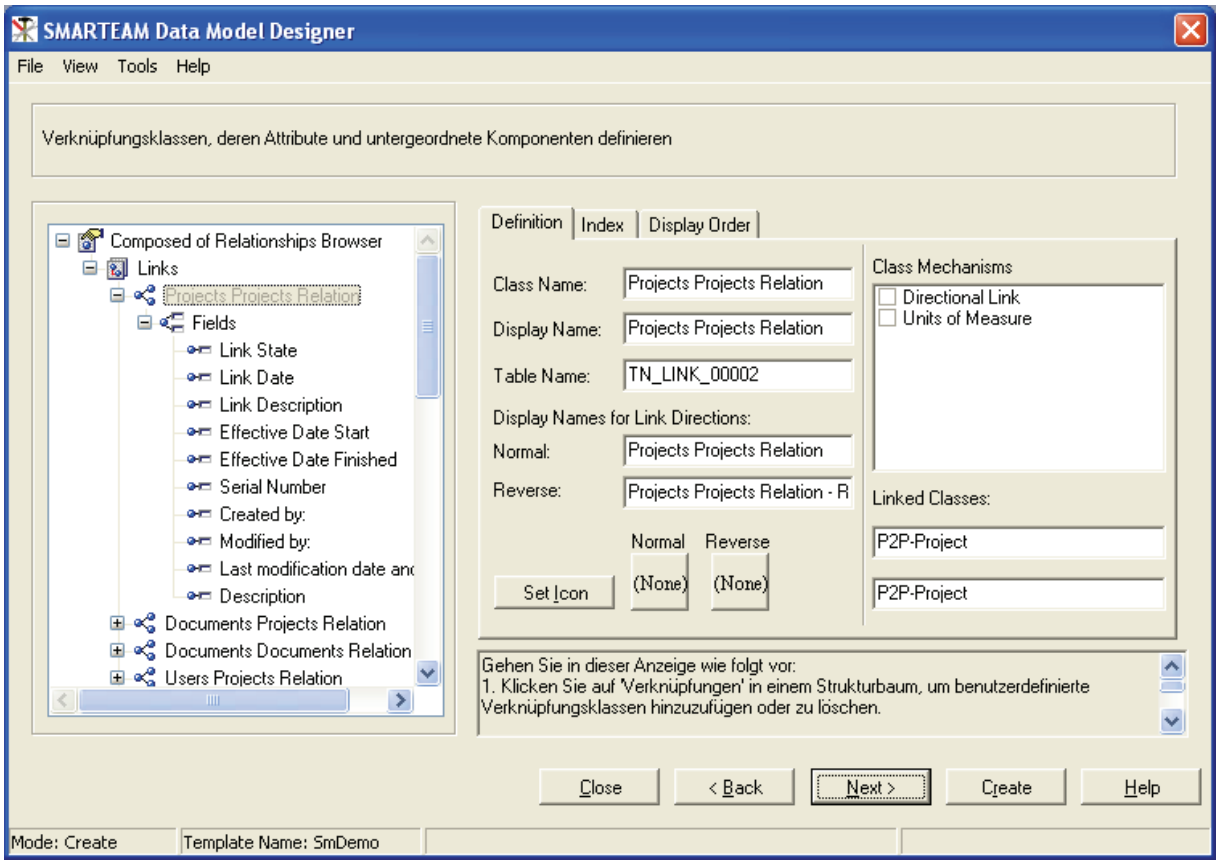


Abbildung 5-34: Definition der Beziehung mit dem SmarTeam Data Model Designer

4. Definition der Profilkarten

Die Definition der Art, wie die Objektinformationen für den PDM-Anwender aufbereitet werden, stellt den letzten Schritt bei dem Überarbeiten des SmarTeam-Datenmodells dar. Die Objektinformationen werden in SmarTeam in Form von klassenspezifischen Profilkarten visualisiert. Diese Profilkarten bilden ein zentrales Element der Benutzeroberfläche. Sobald in SmarTeam ein bestimmtes Objekt bearbeitet wird, wird die Profilkarte der dazugehörigen Klasse angezeigt. Die Eingabe von Attributen erfolgt ebenfalls über die Profilkarten.

Die Definition der Profilkarten erfolgt interaktiv über den Data Modell Designer. In einem grafischen Editor können die Attributfelder halbautomatisch auf der Profilkarte angeordnet und frei manipuliert werden (siehe Abbildung 5-35).

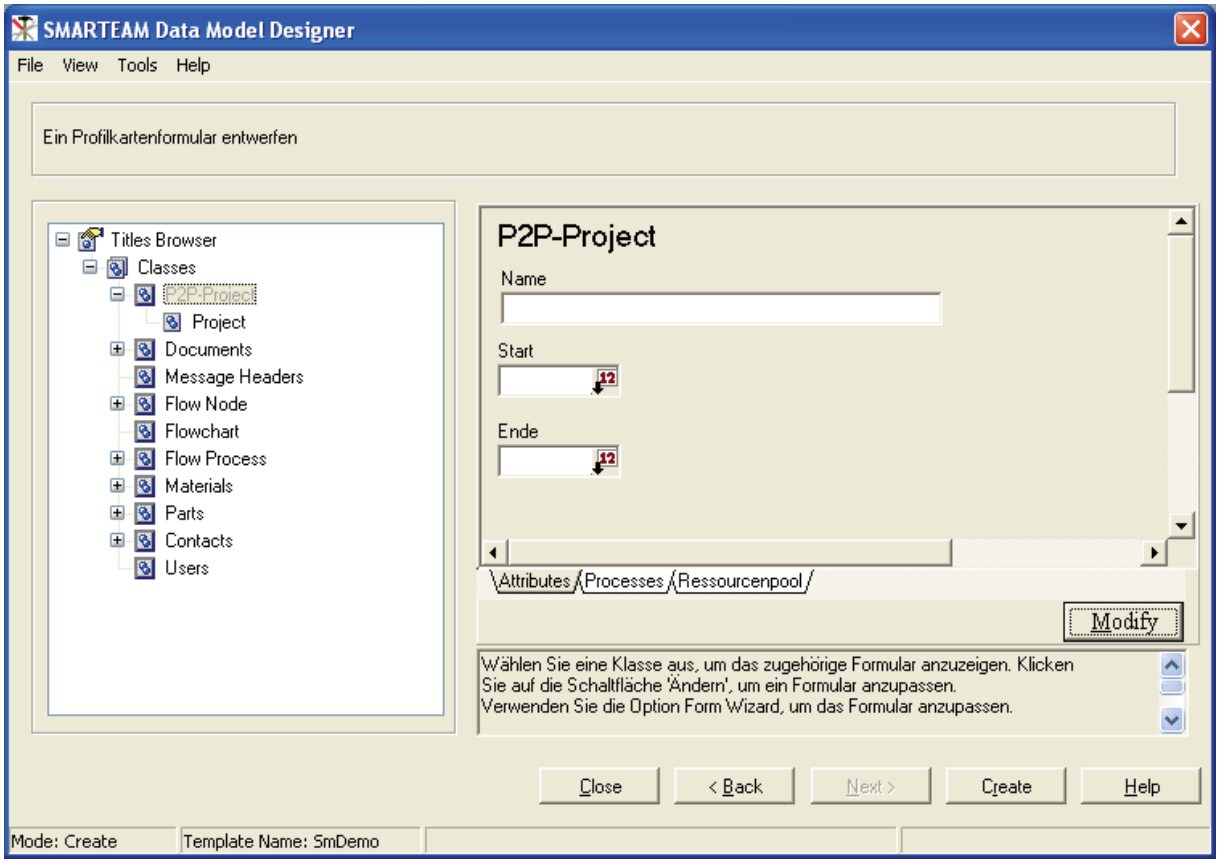


Abbildung 5-35: Definition der Profilkarten mit dem SmarTeam Data Model Designer

Die Definition des kompletten Datenmodells beinhaltet neben den vorgestellten Schritten eine Vielzahl weiterer Teilaufgaben. So ist beispielsweise eine automatische Verknüpfung von Eingabefeldern mit Standardwerten oder Systemvorgaben möglich. Derartige Routineaufgaben zur Konfiguration des Datenmodells können jedoch aus Zeitgründen an dieser Stelle nicht vertieft werden. Nähere Informationen zu diesem Thema können aus der SmarTeam-Dokumentation entnommen werden. [5]

5.6.2 Logikschicht

Der Datenaustausch zwischen der P2-Plantafel und den verwendeten PDM-System(en) wird durch den ***PDM-Adapter*** der Logikschicht abgewickelt. Aus diesem Grund bildet dieses Modul die Schlüsselkomponente für die PDM-PM-Integration.
Um eine hohe Flexibilität zu gewährleisten, ist der PDM-Adapter in zwei Schichten unterteilt (vgl. Abbildung 5-36).

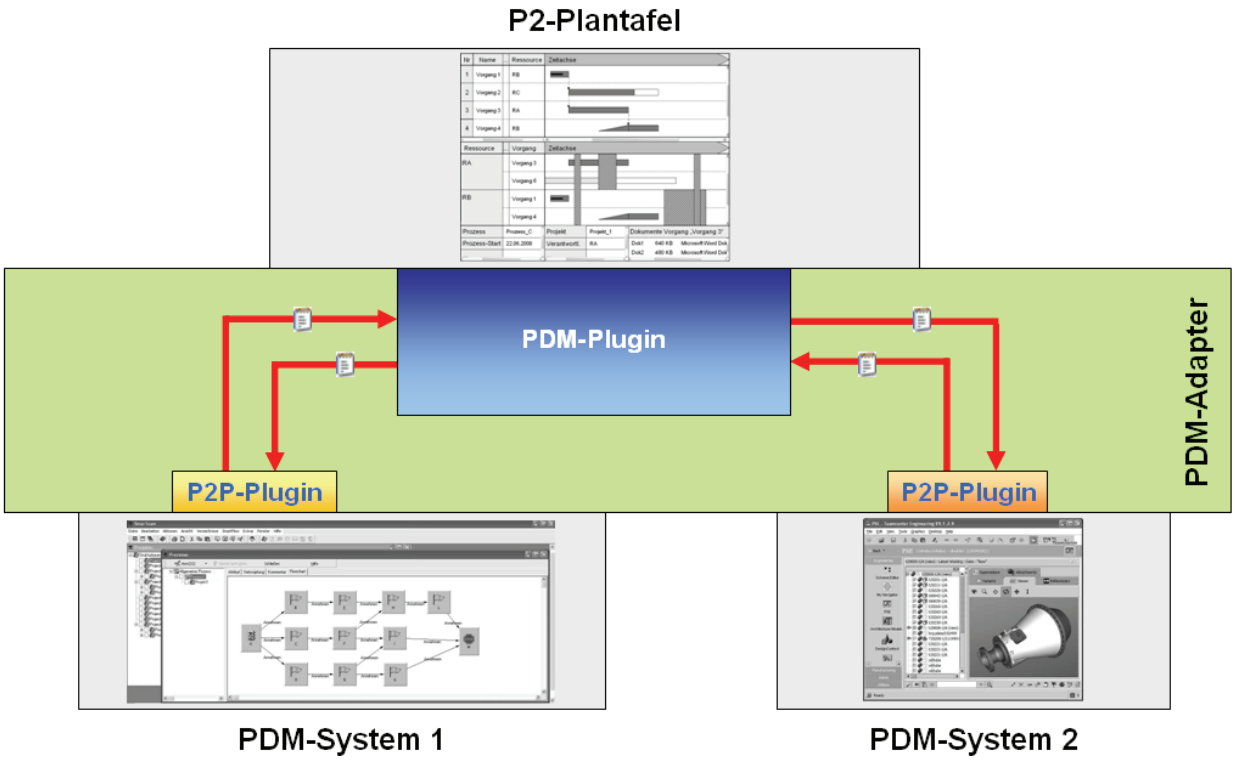


Abbildung 5-36: Architektur des PDM-Adapters

Die obere Schicht des PDM-Adapters dient der Einbindung von PDM-Daten in die Plantafel. Daher wird diese Schicht im Folgenden als ***PDM-Plugin***[41] bezeichnet. Währenddessen stellt die untere Adapterschicht (***P2P-Plugin***) die auf der PDM-Seite benötigten Integrationsroutinen bereit. Dieser Bestandteil des PDM-Adapters ist im Kontext jedes jeweiligen PDM-Systems zu implementieren. Der Datenaustausch zwischen dem PDM-Plugin und verfügbaren P2P-Plugins wird mittels entsprechender Kommunikationsmechanismen realisiert.

Diese Architektur bietet deutliche Vorteile gegenüber konventionellen Schnittstellen. Da das ***PDM-Plugin*** ausschließlich PDM-neutrale Routinen zum Datenaustausch definiert, ist es von dem verwendeten PDM-System unabhängig. Daher ist zur Kombination der P2-Plantafel mit

[41] Plugin: <engl. anschließen> Computerprogramm, das als Erweiterung in ein bestehendes Programm eingebunden wird

einem konkreten PDM-System lediglich eine Erweiterung des ***PDM-Adapters*** um ein ***P2P-Plugin*** für das betreffende PDM-System erforderlich. Dadurch kann die Implementierung für die Integration der P2P mit dem PDM-System SmarTeam unter geringem Aufwand auf andere PDM-Systeme portiert werden.
Weiterhin ermöglicht diese Architektur einen Datenaustausch unabhängig vom verwendeten Betriebssystem. Die Windows-basierte P2P kann dadurch beispielsweise auch mit Linux- und Unix-basierten PDM-Systemen kombiniert werden.
Das Konzept für die P2-Plantafel ist folglich auf die gesamte Breite der verfügbaren PDM-Systeme anwendbar.

Für die Realisierung des PDM-Adapters ist zunächst die Definition der Schnittstelle zwischen den beiden Adapterschichten erforderlich. Die Kommunikation zwischen der PDM- und dem P2P-Plugin erfolgt auf Basis einer Dateischnittstelle. Dabei übertragen die Plugins Befehle und Informationen in Form einer ASCII-Datei in ein vordefiniertes Austauschverzeichnis. Für die beiden möglichen Richtungen des Informationsflusses werden folgende Dateinamen für die Austauschdatei festgelegt. (Die Endung "xf" steht dabei für Exchange File).

- Datei zum Datenaustausch von P2P nach PDM: p2p_pdm.xf
- Datei zum Datenaustausch von PDM nach P2P: pdm_p2p.xf

Das Austauschverzeichnis wird von beiden Plugins auf Dateien des Partnerprogramms überwacht. Das Anlegen und Auslesen der Austauschdateien erfolgt mittels der Klassen ***CListener*** und ***CProfile***. Diese Klassen beinhalten die Basisfunktionalität, die für eine plattformunabhängige Kommunikation erforderlich sind.

Die Klasse **CProfile** ermöglicht die Erstellung von Objekten, die in der Lage sind, strukturierte Informationen in Form von Austauschdateien zu erzeugen und zu interpretieren. Die Klasse **CListener** dient hingegen der Auswertung von eingegangenen Austauschdateien. Objekte dieser Klassen sind in der Lage das Eintreffen einer neuen Datei zu erkennen und die entsprechende Reaktion zu initialisieren.
Da für die P2P und das PDM-System anwendungsspezifische Befehlsbehandlungen erforderlich sind, müssen innerhalb jedes Plugins eigene spezifische Profile- und Listener-Klassen implementiert werden. CListener und CProfile sind daher als virtuellen Basisklassen deklariert. [5]

Eine Austauschdatei hat den folgenden Aufbau:

- Schlüssel (zwingend erforderlich)
- Attribute (optional)
- Listen-Tag(s) und Liste(n) (optional)

Schlüssel

Jede Austauschdatei beginnt mit einem Schlüsselwort, welches die Schnittstellenoperation kennzeichnet. Bei den Schlüsselworten ist zwischen Kommandos und Antworten zu unterscheiden.

Für jedes Kommando muss in der Befehlsbehandlung des Empfängers eine Bearbeitungsroutine vorhanden sein, damit das Kommando ausgeführt wird. Nach Ausführen des Kommandos erfolgt, falls es erforderlich ist, eine Antwort an das aufrufende Programm. Antworten werden von Kommandos durch das Präfix "RE" unterschieden. Für den Fall, dass ein unbekanntes Kommando empfangen wird, wird eine Fehlermeldung ausgegeben.

Zum Auslesen der Projektinformationen aus dem PDM-System schickt die P2-Plantafel beispielsweise das Kommando GET_PROJECT_DATA, woraufhin das PDM-System die Antwort RE GET_PROJECT_DATA sendet.

Attribute

Zusätzliche Informationen, die zur Befehlsausführung notwendig sind, werden als Attribute übergeben. Ein Attributeintrag besteht aus Namen und Wert des Attributs. Als Trennzeichen zwischen diesen beiden Elementen dient das Pipe-Zeichen (|).

Als zentrale Attribute beim Datenaustausch zwischen der P2P und dem PDM-System sind beispielsweise die SmarTeam-Klassen-ID (CID) und SmarTeam-Object-ID (OID) zu übergeben.

Listen-Tag(s) und Liste(n)

Der Austausch von strukturierten Informationen wird mit Hilfe von Listen ermöglicht. Jede Liste wird durch ein Listen-Tag[42] eingeleitet. Ein Tag besteht aus einem Namen der von zwei Underline-Zeichen eingerahmt ist (_). Es dient dazu die nachfolgenden Zeilen als Elemente einer bestimmten Liste zu identifizieren. Eine Liste kann beliebig viele Listenzeilen enthalten, wobei jede Zeile mindestens ein Listenelement beinhaltet. Wenn eine Zeile mehrere Listenelemente enthält, so werden diese durch Pipe-Zeichen voneinander getrennt.

Die folgende Abbildung 5-37 verdeutlicht den prinzipiellen Aufbau einer Austauschdatei:

[42] Tag: <engl.> Markierung oder Auszeichnung

SCHLÜSSEL
ATTRIBUT1|WERT1
ATTRIBUT2|WERT2
ATTRIBUT3|WERT3
ATTRIBUT4|WERT4
LISTENTAG1
WERT1|WERT2|WERT3
WERT1|WERT2|WERT3|WERT4|WERT5
WERT1
WERT1|WERT2|WERT3
LISTENTAG2
WERT1|WERT2|WERT3|WERT4
WERT1
WERT1
WERT1|WERT2

Abbildung 5-37: Prinzipieller Aufbau einer Austauschdatei

Zur Veranschaulichung des Datenaustausches zwischen P2-Plantafel und PDM-System wird im Folgenden der Ablauf des verwendeten Kommunikationsverfahrens anhand eines konkreten Beispiels dargelegt. Abbildung 5-38 veranschaulicht den Prozess des Ladens eines Projektes aus dem PDM-System (PDMS) in die P2P.

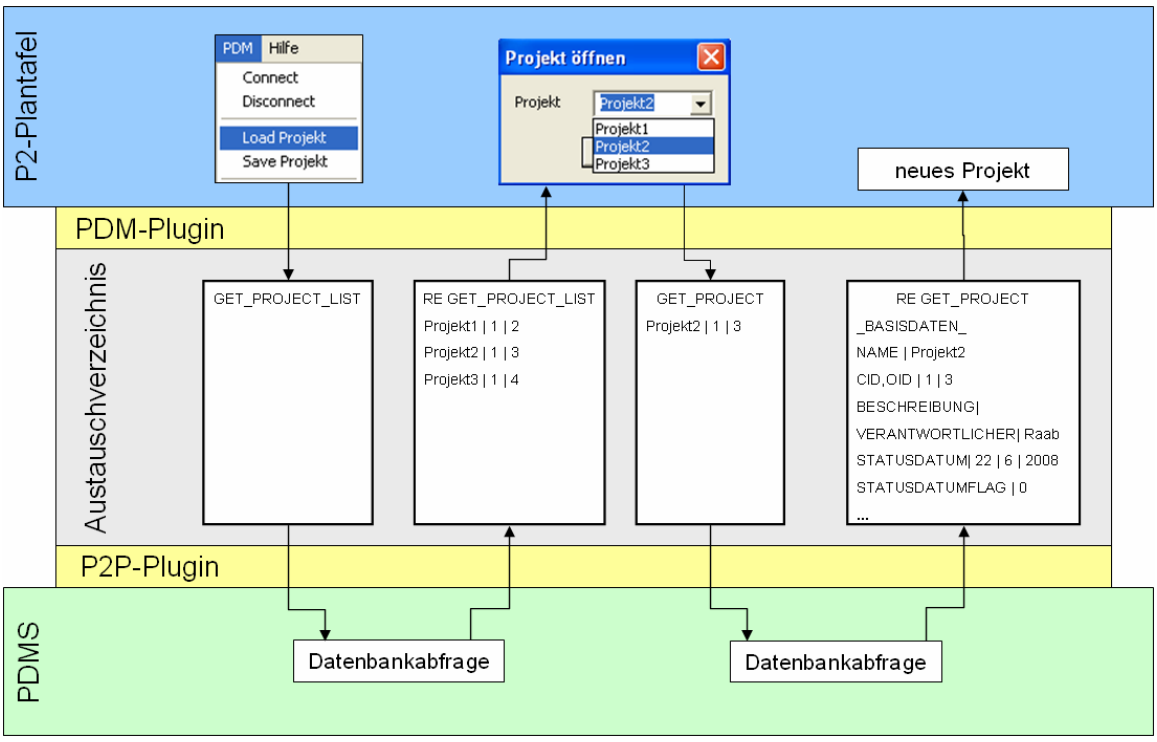


Abbildung 5-38: Kommunikation beim Ladevorgang

Zum Laden eines Projektes ist der Austausch von mehreren Exchange-Files erforderlich. Nachdem in der P2P die Funktion "Laden" aufgerufen wurde, wird zunächst eine Liste der verfügbaren Projekte aus dem PDM-System abgefragt. Die dazu benötigte Austauschdatei besteht lediglich aus dem Schlüssel "GET_PROJECT_LIST".
Nach der Interpretation der Datei durch das in das PDM-System integrierte PDM-Plugin startet dieses eine entsprechende Datenbankabfrage. Für diesen Vorgang werden ausschließlich PDM interne Funktionen verwendet, die über die PDM-API angesprochen werden.
Das Ergebnis dieser Abfrage wird danach in Form eines Exchange-Files an die P2P übermittelt. Die dazugehörige Austauschdatei besitzt die folgende Struktur:

RE GET_PROJEKT_LIST

PROJEKTE
<Projektname1>|<CID>|<OID>
<Projektname2>|<CID>|<OID>
...

Abbildung 5-39: Austauschdatei "RE GET_PROJEKT_LIST"

Die im PDM-System vorhandenen Projekte werden dem Plantafel-Nutzer danach in Form einer Auswahlliste angezeigt. Nach der Wahl des zu ladenden Projektes wird von der Plantafel das Kommando GET_PROJECT abgesetzt. Das dazugehörige XF-File beinhaltet als Attribute den Namen des Projekte sowie seine Klassen- und Objekt-ID (CID und OID). Diese Angaben sind zur eindeutigen Identifizierung des Projektes in der Datenbank des PDM-Systems erforderlich.
Nach Erhalt des Kommandos GET_PROJECT werden von dem PDM-Plugin sämtliche Informationen des betreffenden Projektes aus der PDM-Datenbank ausgelesen. Dies umfasst im Einzelnen die folgenden Angaben:

- die Basisdaten des Projektes,
- den Projektkalender,
- das Leistungsprofil des Projektes
- eine Liste der im Projekt enthaltenen Prozesse,
- die Daten der einzelnen Prozesse,
- den Ressourcenpool des Projektes,
- die Daten der einzelnen Ressourcen sowie
- die Daten der zu den Prozessen gehörigen Vorgängen.

Die folgende Abbildung 5-40 zeigt die Struktur der Austauschdatei die beim Laden eines Projekts erzeugt wird.

RE GETPROJECT

BASISDATEN
NAME|<Projektname>
CID,OID|<int>|<long>
BESCHREIBUNG|<string>
VERANTWORTLICHER|<string>
START,ENDE|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
STATUSDATUM|<tag>|<monat>|<jahr>
STATUSDATUMFLAG|<bool>
ARBEITSSTUNDEN|<int>
WERKTAGE|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>
STANDARDKAPAZITAET|<float>|<int>

PROJEKTKALENDER
<Leerlaufname1>|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
<Leerlaufname2>|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
...

LEISTUNGSPROFIL
KAPAZITAET|<Kapazitaetsname1>|<KapazitaetsFaktor>|<KapazitaetsEinheit>|<EinRuestZeit>
KAPAZITAET|<Kapazitaetsname2>|<KapazitaetsFaktor>|<KapazitaetsEinheit>|<EinRuestZeit>
...

PROZESSE
<Prozessname1>
<Prozessname2>
...

PROZESS: <Prozessname1>
CID,OID|<int>|<long>
BESCHREIBUNG|<string>
START,ENDE|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
VORGANG|<1>|<name>
VORGANG|<2>|<name>
...

PROZESS: <Prozessname2>
CID,OID|<int>|<long>
BESCHREIBUNG|<string>
START,ENDE|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
VORGANG|<1>|<name>
VORGANG|<2>|<name>
...

RESSOURCENPOOL
<Ressourcenname1>|<Art>
<Ressourcenname2>|<Art>
...

RESSOURCE: <Ressourcenname1>
CID,OID|<int>|<long>
WERTE|<int>|<int>|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>
KAPAZITAET|<Kapazitaetsname1>|<KapazitaetsFaktor>|<KapazitaetsEinheit>|<EinRuestZeit>
KAPAZITAET|<Kapazitaetsname2>|<KapazitaetsFaktor>|<KapazitaetsEinheit>|<EinRuestZeit>
...
VORGANG|<Vorgangsname1>|<zugehörigerProzess>|<zugehörigesProjekt|<CID>|<OID>
VORGANG|<Vorgangsname2>|<zugehörigerProzess>|<zugehörigesProjekt|<CID>|<OID>
...
<Leerlaufname1>|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
<Leerlaufname2>|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
...

RESSOURCE: <Ressourcenname2>
CID,OID|<int>|<long>
WERTE|<int>|<int>|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>
KAPAZITAET|<Kapazitaetsname1>|<KapazitaetsFaktor>|<KapazitaetsEinheit>|<EinRuestZeit>
KAPAZITAET|<Kapazitaetsname2>|<KapazitaetsFaktor>|<KapazitaetsEinheit>|<EinRuestZeit>
...
VORGANG|<Vorgangsname1>|<zugehörigerProzess>|<zugehörigesProjekt|<CID>|<OID>
VORGANG|<Vorgangsname2>|<zugehörigerProzess>|<zugehörigesProjekt|<CID>|<OID>
...
<Leerlaufname1>|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
<Leerlaufname2>|<tag>|<monat>|<jahr>|<tag>|<monat>|<jahr>
...

VORGANG: <Prozessname1>.<Vorgangsname1>
CID,OID|<int>|<long>
WERTE|<Dauer>|<DauerAbs>|<Masse>|<Fortschritt>|<Ressource>|<Kapazitaet>
DATEN|<FAZ>|<IAZ>|<SAZ>|<FEZ>|<IEZ>|<SEZ>|ERAZ|FAZ0
VORGAENGER|<int>|<int>|<int>|...
NACHFOLGER|<int>|<int>|<int>|...
FLAGS|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>
SUBPROZESS|<string>

VORGANG: <Prozessname1>.<Vorgangsname2>
CID,OID|<int>|<long>
WERTE|<Dauer>|<DauerAbs>|<Masse>|<Fortschritt>|<Ressource>|<Kapazitaet>
DATEN|<FAZ>|<IAZ>|<SAZ>|<FEZ>|<IEZ>|<SEZ>|ERAZ|FAZ0
VORGAENGER|<int>|<int>|<int>|...
NACHFOLGER|<int>|<int>|<int>|...
FLAGS|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>
SUBPROZESS|<string>

...

VORGANG: <Prozessname2>.<Vorgangsname3>
CID,OID|<int>|<long>
WERTE|<Dauer>|<DauerAbs>|<Masse>|<Fortschritt>|<Ressource>|<Kapazitaet>
DATEN|<FAZ>|<IAZ>|<SAZ>|<FEZ>|<IEZ>|<SEZ>|ERAZ|FAZ0
VORGAENGER|<int>|<int>|<int>|...
NACHFOLGER|<int>|<int>|<int>|...
FLAGS|<bool>|<bool>|<bool>|<bool>|<bool>|<bool>
SUBPROZESS|<string>
...

Abbildung 5-40: Austauschdatei "RE GET_PROJECT"

Sobald das XF-File mit den Projektinformationen in das Austauschverzeichnis übertragen ist, wird in der Plantafel ein Projekt mit den angegebenen Attributen erzeugt. Im Anschluss erfolgt eine Prüfung, ob ein Projekt mit identischer Klassen- und Objekt-ID bereits in der Plantafel vorhanden ist. Sollte dies der Fall sein, wird dieses Projekt durch das neue Projekt ersetzt. Ansonsten wird das neue Projekt zu den vorhandenen Projekten hinzugefügt.

Um die Integrität des Datenbestandes der P2P zu gewährleisten, ist es erforderlich, dass nach dem Laden jeden Projektes geprüft wird, ob von dem aktuellen Projekt andere Projekte referenziert werden. Sollte diese Prüfung positiv ausfallen, erfolgt eine Kontrolle, ob die

betreffenden Projekte bereits in der P2P vorhanden sind. Wenn dies nicht der Fall ist, werden die entsprechenden Projekte ebenfalls aus dem PDM-System geladen.
Je nach Projektstruktur kann aus dem Laden eines Projektes der Import einer Vielzahl weiterer Projekte resultieren. Dies ist beispielsweise der Fall, wenn eine Ressource innerhalb einer Reihe unterschiedlicher Projekte eingesetzt wird. Im Hinblick auf die Anforderung der Benutzerfreundlichkeit (vgl. Kapitel 3) wird der P2P-Anwender über die Notwendigkeit des Importes von jedem zusätzlichen Projekt mittels eines Popup-Fensters informiert. Dadurch wird ihm die Möglichkeit gegeben, den Prozess des Ladens abzubrechen. Nach einem Abbruch wird der Ausgangszustand wiederhergestellt.

Bereits dieses einfache Beispiel zeigt, dass komplexe Funktionsroutinen für die Integration von Plantafel und PDM-System erforderlich sind. Nachfolgend werden die Konzepte für die Plugins detailliert, welche die für die Integration erforderliche Kommunikation zwischen den beiden Systemen abwickeln.

PDM-Plugin

Von dem PDM-Plugin werden das Einlesen und das Erzeugen von Austauschdateien auf Seiten der P2-Plantafel realisiert. Die folgende Abbildung 5-41 zeigt den generellen Aufbau des Plugins und seine Schnittstelle zu der PM-Komponente der Plantafel. Aus Gründen der Übersichtlichkeit ist in dem gezeigten Objektdiagramm der Bereich der P2-Plantafel auf zwei Projekt-Objekte reduziert.

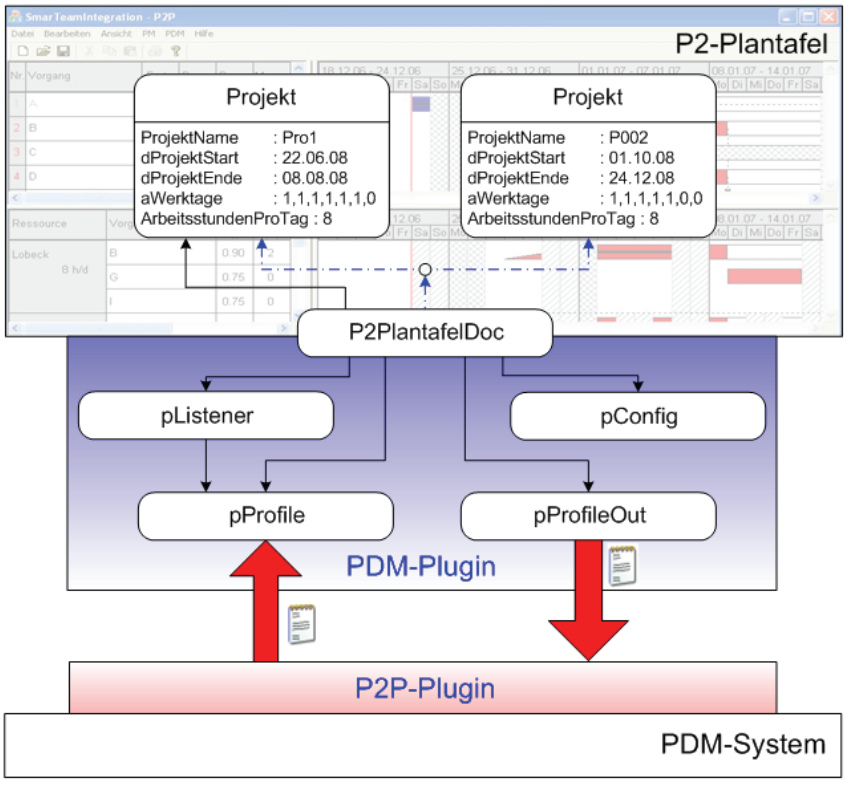


Abbildung 5-41: Architektur des PDM-Plugins (Auszug)

Das Objekt **P2PlantafelDoc** bildet die Schnittstelle zwischen dem Stand-Alone-Modus und dem PDM-Plugin. P2PlantafelDoc referenziert alle vorhandenen Projekte und verfügt über sämtliche Methoden und Funktionalitäten, welche für die Durchführung des Projektmanagements notwendig sind.
Der Datenaustausch zwischen der P2P und einem PDM-System wird mittels Objekten der zuvor beschriebenen Profil- und Listener-Klassen realisiert. P2PlantafelDoc greift auf zwei Profil- sowie ein Listener-Objekt und ein pConfig-Objekt zu.

Das Profil-Objekt **pProfileOut** wird für die ausgehende Kommunikation benötigt. Um beispielsweise in der PDM-Ansicht selektierte Dokumente zu öffnen, wird eine Austauschdatei erzeugt. Die Austauschdatei überträgt das Kommando "OPEN_DOCUMENT" und beinhaltet als Attribute den Namen der Datei sowie ihre IDs (vgl. Abbildung 5-42).

```
OPEN_DOKUMENT
CID | 56
NAME | Dok2
OID | 78
```

Abbildung 5-42: Austauschdatei "OPEN_DOCUMENT"

Die Methode von P2PlantafelDoc zur Erzeugung dieser Austauschdatei ist in Abbildung 5-43 aufgeführt.

```
bool CP2PlantafelDoc::OpenPDMDocument ( Document initDocument )
{
        if ( pProfileOut != NULL )
        {
                pProfileOut->Clear ( );
                pProfileOut->SetCommand (  "OPEN_DOCUMENT"  );
                pProfileOut->AddAttrib(  "NAME", string ( initDocument.Name ) );
                pProfileOut->AddAttrib(  "CID", CTools::IntToString ( initDocument.CID ) );
                pProfileOut->AddAttrib(  "OID", CTools::IntToString ( initDocument.OID ) );
                pProfileOut->Write( );
        }
        else
        {
                AddLogfileLine ("OpenPDMDocument: Error 101");
        }
        return TRUE;
}
```

Abbildung 5-43: Listing der Methode OpenPDMDocument der Klasse CP2PlantafelDoc

Einleitend wird mittels einer if-Abfrage geprüft, ob das Objekt ***pProfileOut*** vorhanden ist. Ist dies nicht der Fall, erfolgt die Ausgabe eine Fehlermeldung. Ansonsten wird zunächst über die Funktion "clear" das Profil zurückgesetzt. Im Anschluss daran wird das Kommando, gefolgt von den Attributen in dem Profil registriert. Da die Funktionen "SetCommand" und

"AddAttrib" als Parameter Zeichenketten (strings) erwarten, ist teilweise eine Konvertierung der übergebenen Parameter durchzuführen. Zu diesem Zweck ist die Hilfsklasse ***CTools*** in die Plantafel eingebunden, welche Funktionen für die Behandlung der benötigten Datentypen bereitstellt.
Mit dem Aufruf der Funktion "write" wird schließlich das Schreiben des Profils in eine ASCII-Datei abgewickelt. Um zu verhindern, dass der P2P-Adapter des PDM-Systems eine unvollständige Datei einliest, wird das Profil zunächst in eine temporäre Datei geschrieben, die erst nach Übertragen des vollständigen Profils in die Austauschdatei ***p2p-pdm.xf*** unbenannt wird.

Die Objekte **pListener** und **pProfile** werden für die Verarbeitung von eingehenden Austauschdateien benötigt. ***pListener*** überwacht das Austauschverzeichnis. Sobald er eine neue Datei mit der Bezeichnung pdm-p2p.xf registriert, aktualisiert er das Profil ***pProfile*** mit den empfangenen Daten. Anschließend ruft ***pListener*** die Methode "PDMFileFound" von CP2PlantafelDoc auf, in der die Auswertung des eingegangenen Profils durchgeführt wird. Abbildung 5-44 zeigt einen Auszug aus dieser Funktion.

```
bool CP2PlantafelDoc::PdmFileFound()
{
	string strCmd = pProfile->GetCommand ( );
	CP2PlantafelDoc::BeginWaitCursor ( );

	if (strCmd == "RE GET_PROJECT_LIST")
	{
		LoadProjectDialog ( );
	}


	else if (strCmd == "RE GETPROJECT")
	{
		SetProject ( );
	}


	else if (strCmd == "RE GET_DOC_LIST")
	{
		SetDocs ( );
	}

	else
	{
		AddLogfileLine ("PDMFileFound: Error 107 - Unknown Command (" + strCmd + ")");
	}

	CP2PlantafelDoc::EndWaitCursor ( );
	pListener->Listen (200);
	return TRUE;
}
```

Abbildung 5-44: Listing der Methode "PdmFileFound" von CP2PlantafelDoc

Für sämtliche von der P2P unterstützen Kommandos, wird in diesem Codesegment in die entsprechende Bearbeitungsroutine verzweigt. Der Eingang des Kommandos "RE GET-_PROJECT_LIST" zieht beispielsweise den Aufruf der Methode "LoadProjectDialog" nach sich. Diese initialisiert die Erstellung eines Dialogfensters, das eine Auswahlliste für alle im PDM-System verfügbaren P2P-Projekte anzeigt. Die dazu erforderlichen Projektnamen werden mittels der Funktion "GetAttrib" aus dem Profil ***pProfile*** ausgelesen.
Damit ***pProfile*** erst nach seiner Auswertung mit neuen Austauschdaten überschrieben werden kann, wird ***pListener*** erst bei Abschluss der Funktion "PDMFileFound" wieder aktiviert ("pListener->Listen (200)").

Das Objekt **pConfig** bildet die letzte Komponente des PDM-Plugins. Mit diesem Objekt wird das flexible Betriebsverhalten des PDM-Plugins realisiert, welches die Kommunikation mit verschiedenen PDM-Systemen ohne Anpassung des Quellcodes ermöglicht. Das Verhalten des P2P-Integrationsmodus lässt sich anhand weniger Steuerungsparameter variieren. Die einzelnen Parameter werden extern in einer Ini-Datei gespeichert und vom Objekt pConfig geladen und ggf. verändert.
Weiterhin werden von pConfig sämtliche Ausgabetexte für das PDM-Plugin verwaltet. Dies ermöglicht einen einfachen Wechsel der Sprachversion des Plugins durch Ersatz der entsprechenden Textpassagen.

P2P-Plugin

Das P2P-Plugin implementiert auf Seite des PDM-Systems die für die Handhabung des Datenaustausches mit der P2-Plantafel erforderlichen Routinen. Abbildung 5-45 verdeutlicht die Architektur der P2P-Plugins.

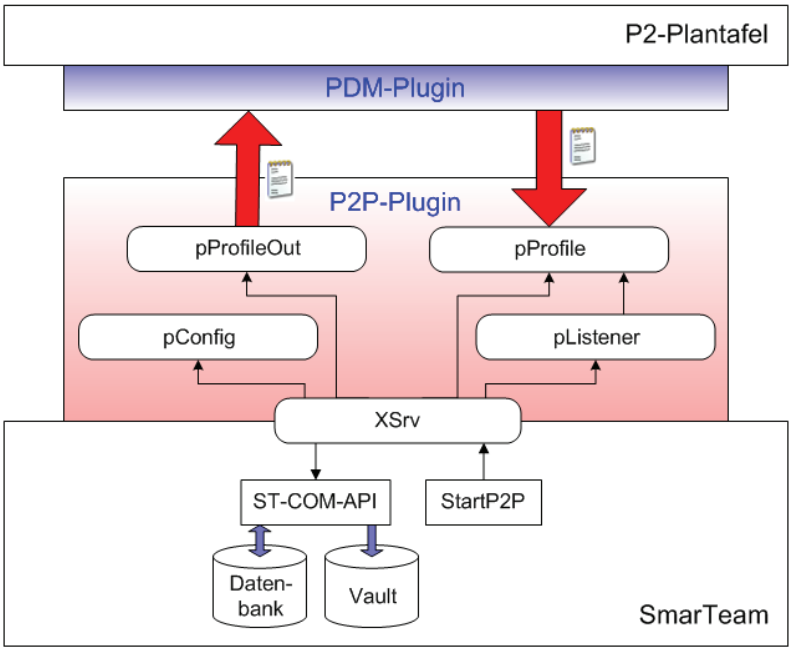


Abbildung 5-45: Architektur des P2P-Plugins

Das P2P-Plugin besitzt einen vergleichbaren Aufbau wie das PDM-Plugin. Das zentrale Objekt des P2P-Plugins ist **XSrv**. Die Aufgabe dieses Objektes besteht darin, der P2-Plantafel die für die PDM-Integration benötigten Funktionen zur Verfügung zu stellen. Dies umfasst im Wesentlichen den vollständigen Zugriff auf die in der PDM-Datenbank hinterlegten Informationen sowie auf die in dem Vault vorhandenen Dokumente. Das Objekt ***XSrv*** ist somit als COM-Server zu verstehen, dessen Dienste von Seiten der Plantafel über Austauschdateien angefragt werden können (XSrv = Exchange-Server).
Das ***XSrv***-Objekt nutzt die gleiche Struktur an Config-, Listener- und Profile-Objekten, wie P2PlantafelDoc um Austauschdateien zu generieren, einzulesen und zu verarbeiten (vgl. Abbildung 5-45 mit Abbildung 5-41). Der Datenaustausch mit dem PDM-System wird derweil über die Objekte und Funktionen der SmarTeam-COM-API abgewickelt (ST-COM-API).
Da das Funktionsprinzip der Austauschdateien bereits hinreichend thematisiert wurde, erfolgt an dieser Stelle eine Beschreibung der API-Einbindung.

Der Datenaustausch über die SmarTeam-API beruht auf der Verwendung von Record-Listen. Diese Datenstruktur besteht aus einem Netzwerk von verketteten Listen, welches dynamisch an die Art und Menge der auszutauschenden Informationen angepasst werden kann.

Da Record-Listen für die SmarTeam-Integration von zentraler Bedeutung sind, ist eine genaue Betrachtung dieser Datenstruktur angebracht. Wie Abbildung 5-46 verdeutlicht, besteht eine Record-Liste aus einer verketteten Liste (Hauptliste), die als Knoten eine endliche Anzahl von verketteten Listen enthält (Sublisten). Jede Subliste trägt dabei eine definierte Art von Information, deren Attribute durch den Kopfknoten (Header) determiniert sind.

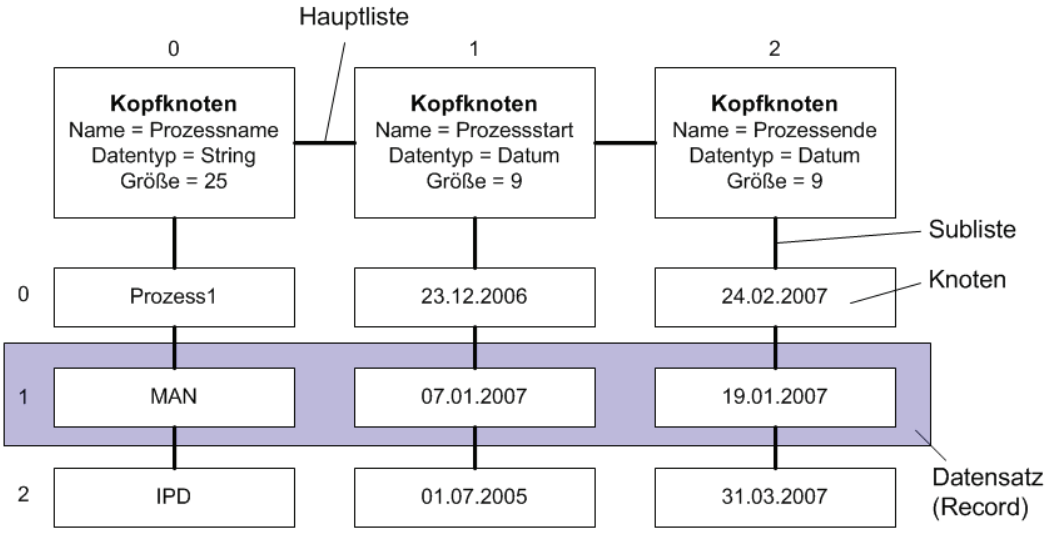


Abbildung 5-46: Record-Liste aus SmarTeam [5] [34]

Beim Speichern eines Datensatzes (Record) werden die einzelnen Sublisten um jeweils einen Knoten erweitert und die Informationen des Datensatzes in die dazugehörigen Knoten eingetragen. Der Zugriff auf die Elemente eines Datensatzes erfolgt derweilen über die Indizierung der verketteten Listen oder die Headerinformationen.
Weiterhin ist die Anzahl der Header einer Record-Liste variabel, so dass zusätzlich zu den vordefinierten Listenstrukturen individuelle Listenstrukturen angelegt werden können.

Abbildung 5-46 zeigt einen Auszug aus einer Record-Liste, welche die Prozesse eines Projektes verwaltet. Die Informationen der Record-Liste können wie eine Tabelle interpretiert werden. Der Header ist dabei als Kopfzeile zu verstehen, die angibt, welche Art von Information in den einzelnen Spalten enthalten ist. Die restlichen Zeilen umfassen die Attribute der einzelnen Datensätze. Somit ist ersichtlich, dass der Prozess 'MAN' beispielsweise eine Laufzeit vom 07.01.07 bis zum 19.01.07 besitzt.

Das Listing in Abbildung 5-47 verdeutlicht anhand einer Funktion des Objektes ***XSrv*** den auf Record-Listen beruhenden Mechanismus des Datenaustausches mit der SmarTeam-Datenbank. Der Zugriff auf die Dokumente des Vaults wird gleichermaßen über die SmarTeam-API realisiert.

Die in Abbildung 5-47 gezeigte Funktion "ChangePDMObject" dient der Änderung von PDM-Objekten. Sie übernimmt als Argumente die Objekt-ID (OID), die Class-ID (CID) und eine Record-Liste (NewAttributes). Die IDs dienen der eindeutigen Identifizierung des zu ändernden Objektes. Währenddessen umfasst die Record-Liste einen einzelnen Datensatz, welcher die neuen Attribute beinhaltet.
Nach dem Laden des Objektes werden über die Funktion "RetrieveObject" die Knoten des ersten Datensatzes der Record-Liste nacheinander durchlaufen. Für jeden Knoten wird zunächst der Headername ermittelt. Danach wird ein Abgleich des Knotennamens mit den Attributsnamen des geladenen Objektes vollzogen. Sobald die beiden Bezeichnungen übereinstimmen, erfolgt, nach einer Kontrolle ob die dazugehörigen Attribute denselben Typ besitzen, die Änderung des betreffenden Attributs des Objektes (Object->PutValue(...)).
Zum Abschluss der Funktion wird schließlich die Aktualisierung des Objektes in der PDM-Datenbank durchgeführt (Object->Update()).

```
bool XSrv::ChangePDMObject(long OID, int CID, ISmRecordListPtr NewAttributes)
{
        long HeaderCount
        HRESULT tRet;
        _bstr_t HeaderName;
        _variant_t Value;
        ISmClassAttributePtr Attribut;

      // Laden des Objekts, das aktualisiert werden soll
        ISmObjectPtr Object=RetrieveObject(CID,OID);

       // Durchlaufen des Datensatzes der Record-Liste mit den neuen Attributen
        HeaderCount=NewAttributes->GetHeaderCount();
        for (int i=0; i<HeaderCount; i++)
        {
          // Ermittlung des Header-Namens und des Wert des dazugehörigen Attributs
            HeaderName=NewAttributes->GetHeaderName(i);
            Value=NewAttributes->GetValue(HeaderName,0);

          // Kontrolle ob das Attribut bei dem Objekt  vorhanden ist
            Attribut=Object->GetAttributes()->ItemByName(HeaderName);
            if (Attribut)
            {
                // Kontrolle ob ValueTypes übereinstimmen
                SmDataTypesEnum Typ1,Typ2;
                Typ1=Object->GetAttributes()->ItemByName(HeaderName)->GetAttributeType();
                Typ2=NewAttributes->GetValueType(HeaderName);

                if (Typ1==Typ2)
                {
                  // Änderung der Objektwerte
                    Object->PutValue(HeaderName,Value);
                }
            }
        }

      // Aktualisierung der Datenbank
        tRet=Object->Update();
        return TRUE;
}
```

Abbildung 5-47: Listing der Methode "ChangePDMObject" von XSrv

Neben dem Datenaustausch besteht eine weitere Anforderung für eine umfassende Integration darin, dass die Plantafel aus dem PDM-System heraus gestartet werden kann (vgl. Anforderungen, Kap.3). Die Anpassung des Systemverhaltens von SmarTeam erfolgt nicht über die COM-API, sondern über einen Editor. Die Script-Maintenance ermöglicht den einzelnen Klassen des SmarTeam-Datenmodells Visual Basic Skripte zuzuordnen und somit benutzerdefinierten Routinen zu klassenspezifischen Ereignissen zu definieren.

Abbildung 5-48 verdeutlicht, dass jeder Klasse des Datenmodells für vordefinierte Ereignisse VB-Skripte zugeordnet werden können. Dabei ist frei definierbar, ob VB-Skripte ***Vor***, ***Nach*** oder ***Anstatt*** der standardmäßigen Ereignisbehandlung ausgeführt werden.

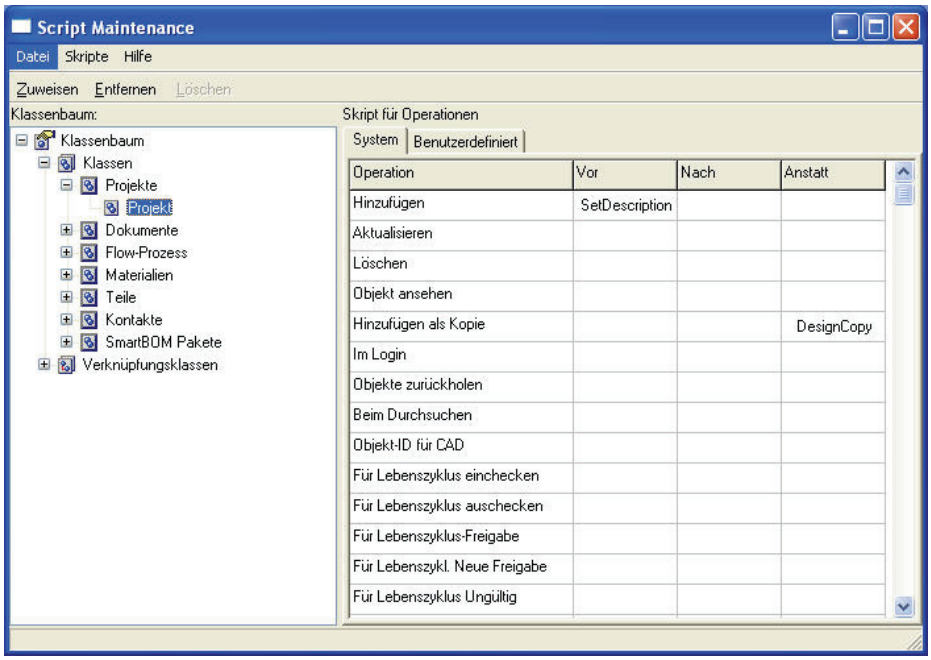


Abbildung 5-48: Einbinden von Erweiterungen mit SmarTeam Script-Maintenance

Die P2-Plantafel wird in die Benutzeroberfläche von SmarTeam mittels des Visual Basic Skriptes ***StartP2P*** integriert. Dieses wird der SmarTeam-Klasse "P2P-Projekt" zugewiesen und anstelle der standardmäßigen Ereignisbehandlung ausgeführt. Das Listing in Abbildung 5-49 zeigt einen Auszug aus dem betreffenden Visual Basic Skript.

```
'Prüfen ob P2-Plantafel bereits aktiv ist
tasks = Process.GetProcessesByName("p2p")
n = tasks.GetLength(0)
If n > 0 Then
   MsgBox("P2P already running !")
End If
Else
   'Starten der P2-Plantfel
   nCount = 0
   Do Until n = 1 Or nCount > 100
      Process.Start("c:\programme\p2p.exe")
      tasks = Process.GetProcessesByName("p2p")
      n = tasks.GetLength(0)
      nCount = nCount + 1
   Loop

   'Initialisierung des COM-Servers XSrv
   P2PMain = CreateObject("xsrv")
   If Not P2PMain.bInit Then
      MsgBox("Error: COM-Server initialisation failed")
   End If

   'Auswertung des Aufrufes
   If sStr = "EDIT" Then
      P2PMain.OpenProject(StrClassId, StrObjectId)
   ElseIf Not sStr = "START" Then
      MsgBox("Error: Undefined Operation")
   End If
End If
```

Abbildung 5-49: Listing des Visual Basic Skriptes "StartP2P" (Auszug)

Das dargestellte Visual Basic Programm prüft zunächst, ob bereits eine Instanz der P2-Plantafel aktiv ist. Sollte dies der Fall sein, wird der Anwender über eine Fehlermeldung darauf aufmerksam gemacht, dass bereits eine lokale P2P-Sitzung läuft und danach die Ausführung des VB-Skriptes beendet.
Andernfalls wird die Plantafel durch Aufruf der Funktion "Process.Start( )" gestartet und danach das COM-Server-Objekt "XSrv" initialisiert. Dieses ist innerhalb des Skriptes über das Objekt "P2PMain" ansprechbar. Das XSrv-Objekt wird erst beim schließen der P2-Plantafel wieder zerstört. Dadurch ist sichergestellt, dass das P2P-Plugin während der kompletten Laufzeit der P2P in der Lage ist, Anfragen zu empfangen, zu verarbeiten und die benötigten Antworten zu schicken.
Im Anschluss an den Start des COM-Servers wird geprüft, ob die Plantafel über das Ereignis "Start" oder "Edit" aufgerufen wurde. Das Ereignis "Edit" öffnet das aktive Projekt in der Plantafel durch Aufruf der Server-Funktion "OpenProjekt", während das Ereignis "Start" des Startens einer leeren Plantafel dient und daher keine weiteren Operationen erfordert.

Da die Durchführung des Projektmanagements ausschließlich über die P2P-Plantafel erfolgt, ist nach dem Starten der Plantafel keine vom PDM-System initialisierte Kommunikation mehr erforderlich.

5.6.3 Präsentationsschicht

Der wesentliche Teil des Konzeptes zur Datenrepräsentation wurde im Rahmen der Gestaltung des Stand-Alone-Modus ausgearbeitet (vgl. Kapitel 5.5.3). Aus diesem Grund ist für den Integrationsmodus lediglich die Erweiterung des bisherigen Konzeptes um eine PDM-spezifische Ansicht durchzuführen.
Die folgende Abbildung 5-50 illustriert die für den Integrationsmodus entwickelte Ansicht.

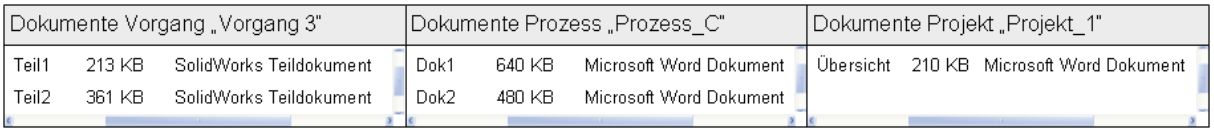


Abbildung 5-50: Ansicht 5: PDM-Ansicht (Integrationsmodus)

Die PDM-Ansicht zeigt für den aktiven Vorgang, den aktiven Prozess sowie das aktive Projekt jeweils eine Liste der dazugehörigen Dokumente. Neben den Namen der einzelnen Dokumente können optional eine Reihe von Zusatzinformationen, wie zum Beispiel Art und Größe des Dokumentes, eingebunden werden.
Zudem werden aus Gründen der Übersichtlichkeit in der Kopfzeile der PDM-Ansicht die Namen des aktiven Vorgangs, Prozesses und Projektes angezeigt.
Weiterhin kann durch Selektion eines Dokuments der Zugriff auf das betreffende Dokument erfolgen. Dazu werden, über die in der Logikschicht dargelegten Mechanismen, die für das Öffnen von Dokumenten verwendeten Routinen des PDM-System eingebunden.

Die PDM-Ansicht wird weiterhin dazu benutzt, um dem Anwender ein direktes Feedback darüber zu geben, in welchem Betriebsmodus sich die P2-Plantafel befindet. Sobald die Verbindung zum PDM-System getrennt wird, ändert sich die Kopfzeile der PDM-Ansicht zu "PDM-Integration deaktiviert" (vgl. Abbildung 5-51).

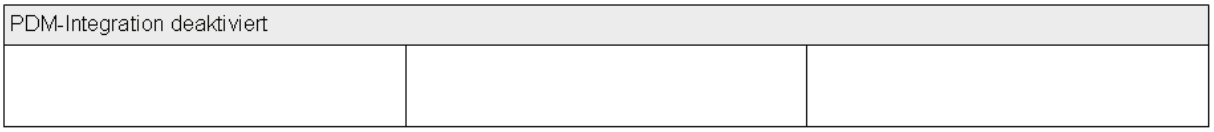


Abbildung 5-51: Ansicht 5: PDM-Ansicht (Stand-Alone-Modus)

Abbildung 5-52 zeigt die PDM-Ansicht noch einmal im Kontext der gesamten Plantafel-Oberfläche.

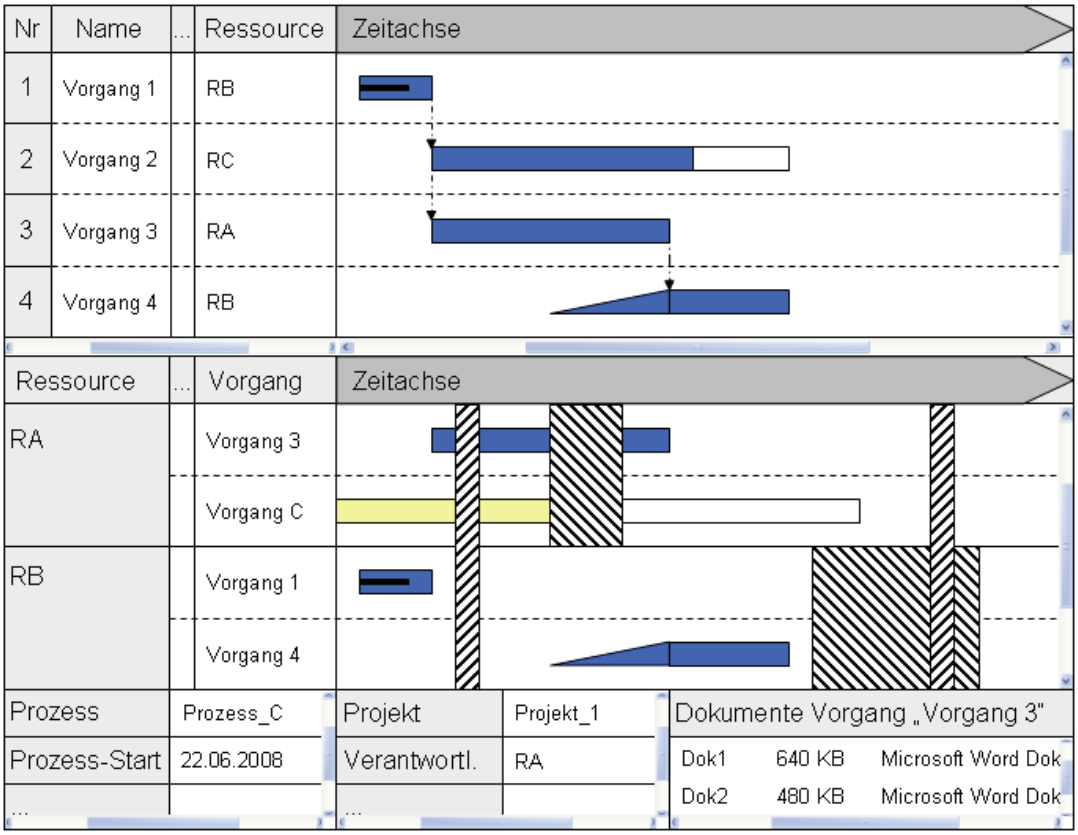


Abbildung 5-52: Plantafel Oberfläche

Die Präsentationsschicht des Integrationsmodus komplettiert das im Rahmen dieser Dissertation erarbeitete Konzept zur Optimierung von Produktentwicklungsprojekten.

Nachfolgend wird die Anwendung des Konzeptes anhand einer prototypenhaften Realisierung vorgestellt.

6 Prototypenhafte Realisierung

In diesem Kapitel wird die Anwendbarkeit des Konzeptes für ein integriertes Produktdaten- und Projektmanagement anhand der prototypenhaften Implementierung der P2-Plantafel im Kontext des PDM-Systems SmarTeam nachgewiesen.
Da die Darlegung des kompletten Umfanges des erstellten Prototypen den Rahmen dieser Dissertation sprengen würde, beschränkt sich dieses Kapitel auf die Beschreibung der zentralen Komponenten der P2-Plantafel sowie der Integration der Plantafel in das PDM-System SmarTeam.

6.1 P2-Plantafel

Die P2-Plantafel wurde entsprechend den in Kapitel 3 formulierten Anforderungen und der in Kapitel 5.3 konzeptionierten Integrationsstrategie in Form einer eigenständigen Windows-Anwendung realisiert. Gemäß dem Klassenkonzept aus Kapitel 5.5.2 wurden dafür spezielle Anwendungsklassen erstellt, die auf dem MFC-Framework beruhen. Abbildung 6-1 zeigt den Prototyp der P2-Plantafel mit einem fiktiven Projekt.

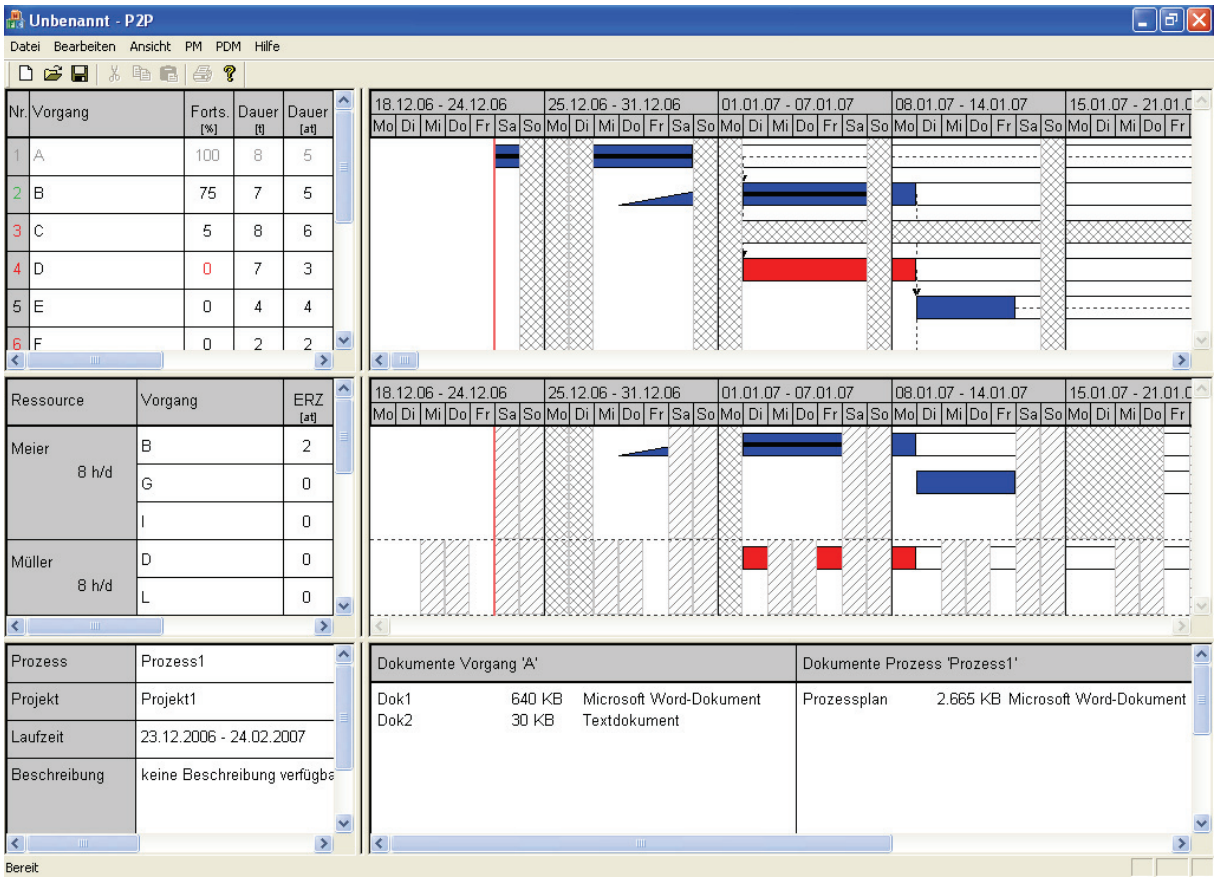


Abbildung 6-1: Benutzeroberfläche der P2-Plantafel

Die Hauptansicht der Benutzeroberfläche der P2P setzt sich zusammen aus einem ***Gantt-Diagramm*** (oben), einem ***Ressourcendiagramm*** (Mitte), der ***Prozess-Ansicht*** (unten, links) und der ***PDM-Ansicht*** (unten, rechts). Die Höhe und Breite der einzelnen Ansichten können per Drag&Drop beliebig skaliert werden.

Die ***Projektansicht*** und zwei zusätzliche ***tabellarische Ansichten*** sind standardmäßig ausgeblendet. Dies ist darin begründet, dass die Informationen dieser Ansichten für die meisten Aspekte das Projektmanagement nicht relevant sind. Aus Gründen der Ergonomie werden sie daher zugunsten von breiteren Balkendiagrammen ausgeblendet. Die unterdrückten Ansichten können bei Bedarf eingeblendet und ebenfalls per Drag&Drop skaliert werden (vgl. Abbildung 6-2).

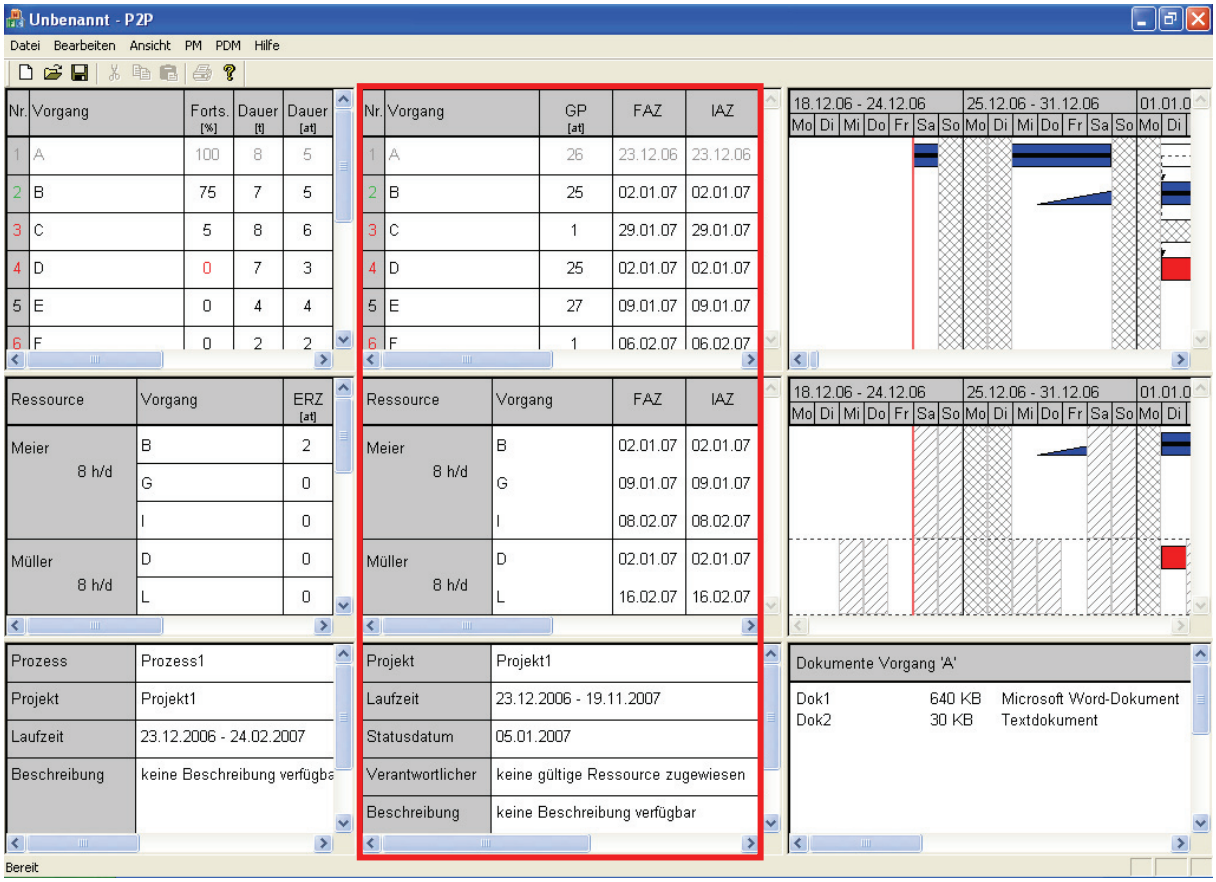


Abbildung 6-2: Benutzeroberfläche der P2-Plantafel

Die einzelnen Ansichten werden nachfolgend kurz vorgestellt.

Gantt-Diagramm

Die Hauptaufgabe des Gantt-Diagramms besteht in der übersichtlichen Aufbereitung sämtlicher Informationen zu dem aktiven Prozess.

Bei dem realisierten Prototypen setzt sich das Gantt-Diagramm aus drei Teilansichten zusammen.

Im grafischen Teil des Gantt-Diagramms, dem sogenannten ***Balkendiagramm***, erfolgt die Abbildung der zeitlichen Lage der einzelnen Vorgänge sowie der Beziehungen zwischen den Vorgängen. Die P2-Plantafel verwendet für die Hervorhebung von unterschiedlichen Informationen verschiedene Formen, Farben und Schraffuren.
Die zeitliche Lage eines Vorganges wird durch einen rechteckigen Balken dargestellt, dessen Position und Länge durch die eingeblendete Zeitskala determiniert sind. Der Vorgangsfortschritt wird als schmaler, schwarzer Balken in jeden Vorgang eingetragen, während Puffer- und Pseudopufferzeiten[43] als weiße bzw. schraffierte Balken an den jeweiligen Vorgang angrenzen. Weiterhin werden die Leerlauftage aus dem Projektkalender in Form von vertikalen, schraffierten Balken visualisiert und die Beziehungen zwischen den Vorgängen als Pfeile dargestellt. Zudem wird jeder Vorgang mit einem Dreieck beginnend abgebildet, sobald er mit einer Einarbeitungs- oder Rüstzeit belegt ist. Die Breite des Dreiecks entspricht der Dauer der Einrüstzeit.

Das Gantt-Diagramm ermöglicht durch Kombination des Balkendiagramms mit einer tabellarischen Ansicht die effiziente Auswertung einzelner Vorgänge und eine umfassende Übersicht über den Gesamtprozess. In der Benutzeroberfläche des Prototypen werden die zentralen Vorgangsinformationen des aktiven Prozesses in der sogenannten ***Übersichttabelle*** dargestellt. Diese setzt sich zusammen aus dem Vorgangsfortschritt in Prozent, der Vorgangsdauer in Tagen und Arbeitstagen, der Masse in der jeweiligen Masseneinheit sowie der benötigten Kapazität und der zugeteilten Ressource (vgl. Abbildung 6-1, links).

Das Balkendiagramm bildet die Lage der einzelnen Vorgänge und die Beziehungen zwischen den Vorgängen ab. Eine andere Möglichkeit für die Wiedergabe dieser Informationen besteht in der Verwendung einer tabellarischen Darstellung. Während die grafische Darstellung einen umfassenden Überblick über den kompletten Prozess ermöglicht, können anhand der tabellarischen Darstellung gezielt einzelne Vorgänge ausgewertet werden. Bei dem realisierten Prototypen erfolgt daher die Angabe der Pufferzeit, der Vorgangsbeziehungen und der Termine der Vorgangskenngrößen (FAZ, SAZ, etc.) sowohl in dem Balkendiagramm als auch in einer zusätzlichen Tabelle. Diese Ansicht wird als ***Plantabelle*** bezeichnet und ist zwischen der zuvor diskutierten Übersichttabelle und dem Balkendiagramm angeordnet (siehe Abbildung 6-2, Mitte).

Durch die verschiedenen Kombinationsmöglichkeiten der drei Segmente des Gantt-Diagramms ist sichergestellt, dass diese Ansicht flexibel an die Bedürfnisse eines Anwenders oder an eine spezifische Planungssituation angepasst werden kann.

---

[43] Pseudopuffer: Pufferzeit die effektiv nicht genutzt werden kann, weil die zugewiesene Ressource in diesem Zeitraum einen Leerlauf hat

Die Standardeinstellung der Benutzeroberfläche des Prototypen zeigt die klassische Form des Gantt-Diagramms, die sich aus der Übersichttabelle und dem Balkendiagramm zusammensetzt. Die Plantabelle ist in dieser Einstellung ausgeblendet (siehe Abbildung 6-3).

Abbildung 6-3: Gantt-Diagramm (Standardansicht)

Wie in Abbildung 6-3 zu erkennen ist, werden zur Steigerung der Benutzerfreundlichkeit in dem Gantt-Diagramm der Status eines Vorganges sowie eventuelle Restriktionsverletzungen optisch hervorgehoben. Im Balkendiagramm werden die Vorgänge in den Farben rosa (kritischer Weg), rot (Restriktionsverletzung) und blau (alles in Ordnung) dargestellt. Währenddessen ist in den tabellarischen Ansichten der Vorgangsstatus anhand der Farbe der Vorgangsnummer ersichtlich.

Damit auch Projektmanagement-Einsteiger den Grund für eine Restriktionsverletzung schnell eingrenzen können, wird die Ursache der Restriktionsverletzung in der Tabelle rot hervorgehoben. Wenn beispielsweise der Vorgangsfortschritt hinter der Planung zurückliegt, wird der Wert des Forschritts rot eingefärbt.

Weiterhin kann über die Funktion "Was stimmt nicht" für jeden Vorgang eine Fehleranalyse durchgeführt werden (siehe Abbildung 6-4). Auf diese Weise können Projektengpässe schnell identifiziert und behoben werden.

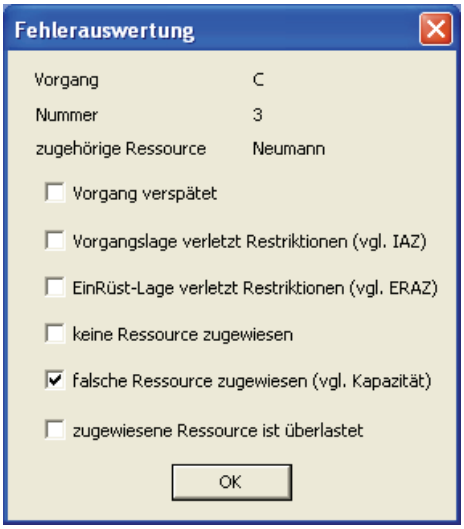


Abbildung 6-4: Fehleranalyse

Die Bearbeitung von Vorgangsattributen erfolgt primär über das in Abbildung 6-5 dargestellte Dialogfenster. Es kann entweder durch Doppelklicken auf einen Vorgang bzw. seine Tabellenzeile oder über das Hauptmenü geöffnet werden. Alternativ können die Vorgänge auch über grafische Eingaben redigiert werden. Dazu muss der Mauszeiger zunächst auf den zu betreffenden Vorgang positioniert werden. Durch anschließende Bewegung der Maus mit gedrückter linker Maustaste kann der Vorgang verschoben werden. Bei horizontaler Bewegung wird die zeitliche Lage des Vorgangs verändert, während bei vertikaler Bewegung die Vorgangsreihenfolge angeglichen wird. Dabei wird die sogenannte Rubber Band Technic verwendet. Dieses Verfahren aktualisiert den Vorgang während des Verschiebens dynamisch.

Abbildung 6-5: Dialogfenster

Für das Anlegen von neuen Vorgängen und die Änderung der Prozessstruktur sind Kontextmenüs in das Gantt-Diagramm eingepflegt. Die Namen der Funktionen des in Abbildung 6-6 dargestellten Kontextmenüs sind selbsterklärend.

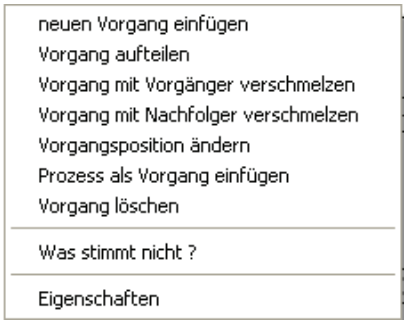


Abbildung 6-6: Kontextmenü zur Vorgangsmanipulation

An dieser Stelle sei außerdem darauf hingewiesen, dass zur Erhöhung der Transparenz verschiedene Zusatzinformationen in das Balkendiagramm eingeblendet werden können. Im Einzelnen handelt es sich dabei um verschiedene Angaben zu den einzelnen Vorgängen in Textform sowie um eine Datumslinie, die das aktuelle Statusdatum anzeigt und somit einen grafischen Soll-Ist-Vergleich realisiert.
Weiterhin werden von den Prototypen drei verschiedene Zeitskalen unterstützt (Tageskala, 3-Tageskala und Wochenskala).

Ressourcendiagramm

Das Ressourcendiagramm bildet die Ressourcenverfügbarkeit und –auslastung ab. Um eine durchgehende Benutzerführung zu gewährleisten, entspricht sein grundlegender Aufbau dem des Gantt-Diagramms.
Im Ressourcendiagramm werden sämtliche für das Projekt verfügbare Ressourcen sowie alle dazugehörigen Vorgänge in alphabetischer Reihenfolge abgebildet (vgl. Abbildung 6-7). Analog zu dem Gantt-Diagramm erfolgt die Aufbereitung der Vorgangsinformationen in Form eines Balkendiagramms sowie von zwei Tabellen. Der Fokus der Übersichttabelle liegt auf der Angabe der ressourcenspezifischen Daten eines Vorgangs, wie z.B. der Einrüstzeit (ERZ), der vorhandenen Kapazitätsgröße, der benötigten Kapazität und des Arbeitspensums, während in der Plantabelle terminliche Vorgangsdaten erfasst werden.

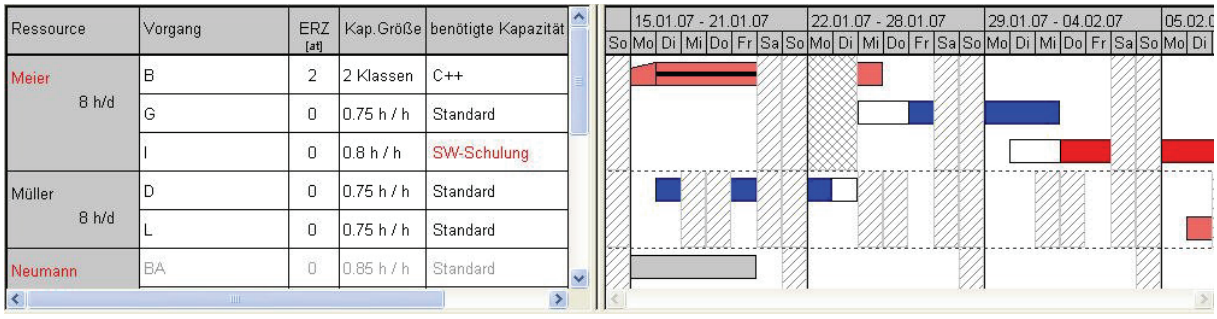


Abbildung 6-7: Ressourcendiagramm (Standardansicht)

Da das Ressourcendiagramm der Planung des Ressourceneinsatzes dient, werden in seinem Balkendiagramm keine Vorgangsbeziehungen angezeigt. Stattdessen werden die Vorgänge des aktiven Prozesses farbig hervorgehoben und die Vorgänge fremder Prozesse grau hinterlegt. Die Vorgänge des aktiven Projektes können mit denselben Techniken wie die Vorgänge des Gantt-Diagramms bearbeitet werden. Der Zugriff auf Vorgänge aus anderen Prozessen ist aus Sicherheitsgründen den Projektverantwortlichen des dazugehörigen Projektes vorbehalten.

Ein weiterer Unterschied zwischen dem Gantt- und dem Ressourcendiagramm besteht darin, dass im Ressourcendiagramm, neben den globalen Leerlauftagen des Projektes, auch die Leerläufe der einzelnen Ressourcen abgebildet werden. Diese werden als schraffierte vertikale Balken dargestellt, wobei die Art des Leerlaufs durch verschiedene Schraffuren gekennzeichnet ist. Leerläufe die durch arbeitsfreie Wochentage bedingt sind erhalten eine einfache Schraffur, während Leerläufe aus dem Projekt- oder Ressourcenkalender durch eine gekreuzte Schraffur hervorgehoben werden. In dem fiktiven Projekt aus Abbildung 6-7 ist beispielsweise ersichtlich, dass die Ressource 'Meier' prinzipiell montags bis freitags arbeitet. Am Montag, den 22.01.07 und Dienstag den 23.01.07 liegen für diese Ressource allerdings Leerlauftage an. Ein Abgleich mit dem Ressourcenkalender verrät, dass es sich bei diesen Leerläufen um Erholungsurlaub handelt.
Weiterhin verfügt das Ressourcendiagramm, zusätzlich zu den vorgangsspezifischen Dialogen und Kontextmenüs, über vergleichbare Bedienelemente für die Erstellung und die Bearbeitung von Ressourcen.

Prozess- und Projektansicht

Die Prozess- und Projektansicht stellen die wesentlichen Informationen zu dem aktiven Prozess bzw. dem aktiven Projekt in tabellarischer Form dar (siehe Abbildung 6-8). Über entsprechende Dialogfelder und Kontextmenüs können die Eigenschaften des Prozesses und des Projektes festgelegt werden sowie Prozesse bzw. Projekte hinzugefügt, gelöscht, geschlossen oder geöffnet werden.

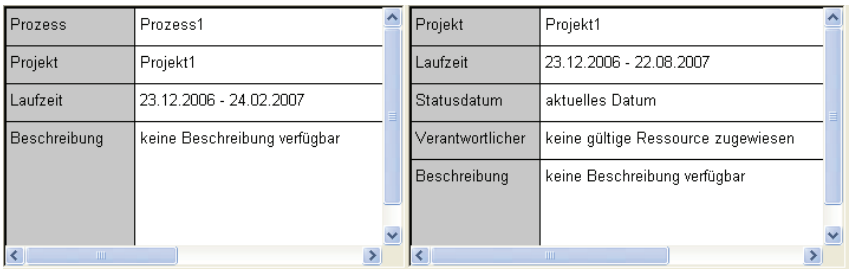


Abbildung 6-8: Prozessansicht (l) und Projektansicht (r)

Um der Projektleitung eine Übersicht über das Gesamtprojekt zu gestatten, unterstützt der entwickelte Prototyp zudem die Anzeige einer Projektübersicht. Diese Ansicht stellt im Gantt-Diagramm alle Prozesse des Projektes dar und bildet zwischen den Prozessen vorliegende Beziehungen ab. Weiterhin werden im Zuge der Terminierung der Gesamtfortschritt der einzelnen Prozesse sowie ihre Gesamtmasse in Arbeitsstunden und ihre Gesamtdauer in Werk- bzw. Kalendertagen berechnet und in der Prozessübersicht dargestellt (vgl. Abbildung 6-9). Zur Steigerung der Benutzerfreundlichkeit ermöglicht die Projektübersicht zudem den direkten Zugriff auf jeden Prozess, durch doppelte Mausselektion (Doppelklicken).

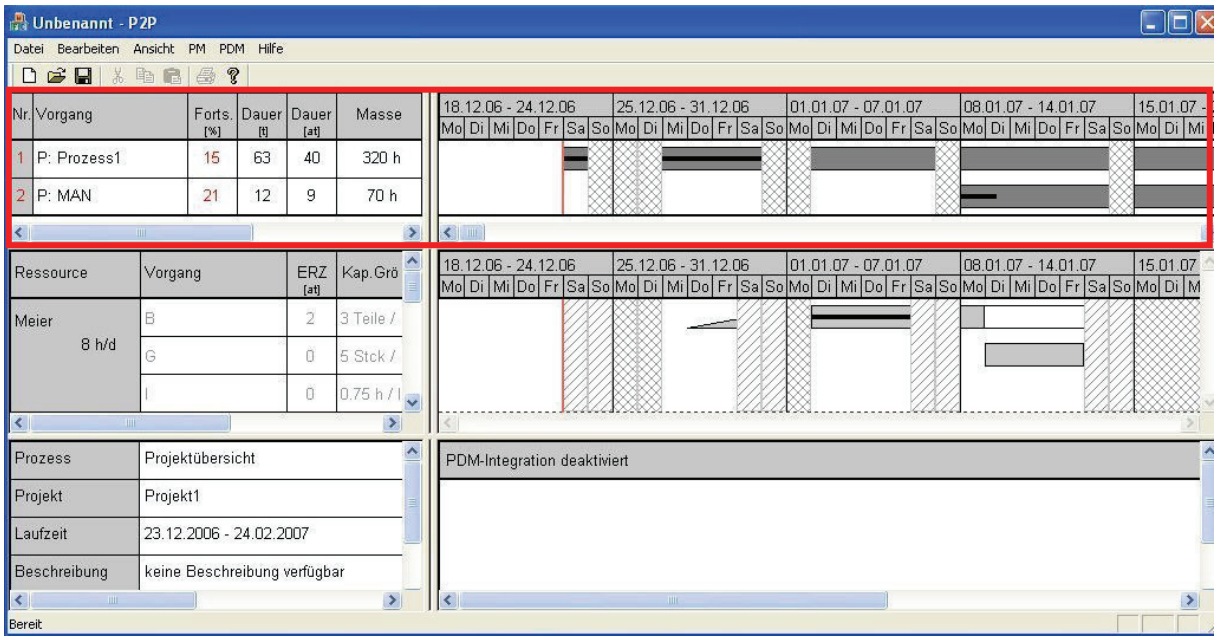


Abbildung 6-9: Projektübersicht

PDM-Ansicht

Die PDM-Ansicht ist die primäre Schnittstelle der P2P zum PDM-System. Sie dient sowohl der Rückmeldung des Verbindungsstatus als auch der Übersicht über die Dokumente des PDM-Systems. Abbildung 6-10 verdeutlicht, dass die PDM-Ansicht zu dem aktiven Vorgang, dem aktiven Prozess und dem aktiven Projekt jeweils die dazugehörigen Dokumente inklusive Dateigröße und Datentyp anzeigt. In diesem Beispiel handelt es sich um die Dokumente des 'Vorgangs A', des Prozesses 'Prozess1' und des Projektes 'Projekt1'

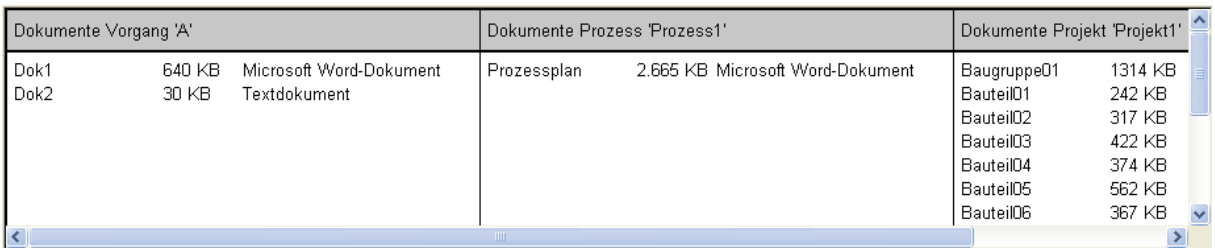


Abbildung 6-10: PDM-Ansicht (Status: PDM-Integration aktiviert)

Der Verbindungsstatus wird über die Kopfzeile der PDM-Ansicht ausgewertet. Für den Fall, dass die PDM-Integration deaktiviert ist und sich die Plantafel im Stand-Alone-Modus befindet, wird anstelle des in Abbildung 6-10 gezeigten Datei-Explorers die Nachricht "PDM-Integration deaktiviert" angezeigt (vgl. Abbildung 6-11).

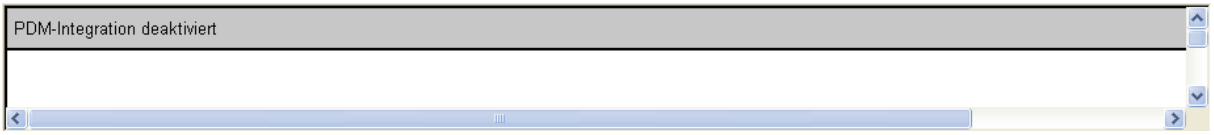


Abbildung 6-11: PDM-Ansicht (Status: PDM-Integration deaktiviert)

Der Datenaustausch mit dem PDM-System wird über den in Kapitel 5.6 ausgearbeiteten Integrationsmechanismus realisiert. Sobald der Anwender ein neues Objekt selektiert, wird vom PDM-Plugin der P2-Plantafel ein XF-File mit den Befehl "GET_DOCS" erzeugt und in ein vordefiniertes Austauschverzeichnis gespielt. Die Austauschdatei wird vom P2P-Plugin des PDM-Systems verarbeitet. Nachdem durch eine Datenbankabfrage die Dokumente ermittelt wurden, die mit den angegebenen Objekten verknüpft sind, erfolgt die Rückgabe dieser Informationen in Form einer zweiten Austauschdatei. Diese trägt den Befehl "RE GET_DOCS". Die Informationen aus der Antwort des PDM-Systems werden danach in der PDM-Ansicht abgebildet.
Abbildung 6-12 verdeutlicht diesen Prozess anhand von Auszügen aus den Austauschdateien auf denen der in Abbildung 6-10 gezeigte Screenshot beruht.

GET_DOCS	**RE GET_DOCS**
PROJEKT|Projekt1|1|67 PROZESS|Prozess1|2|43 VORGANG|A|3|09	_PROJEKT_ Baugruppe01|12|48|SolidWorks Baugruppe|1314 KB Bauteil01|13|47|SolidWorks Teildokument|242 KB ... Bauteil11|13|49|SolidWorks Teildokument|612 KB Ex1|89|01|Microsoft Excel-Arbeitsblatt|230 KB _PROZESS_ Prozessplan|13|45|Microsoft Word-Dokument|2.665 KB _VORGANG_ Dok1|12|4|Microsoft Word-Dokument|640 KB Dok2|56|98|Textdokument|30 KB

Abbildung 6-12: Austauschdateien

Das Laden und Speichern von Projekten wird über den Transfer von vergleichbaren Austauschdateien verwirklicht. Abbildung 6-13 illustriert den Datenaustausch zwischen der P2-Plantafel und dem PDM-System SmarTeam. Im oberen Teil der Abbildung wird ein in der Plantafel modellierter Prozess gezeigt. Darunter ist derselbe Prozess in SmarTeam dargestellt, nachdem das dazugehörige Projekt von der Plantafel in dem PDM-System gespeichert wurde. Der Vergleich der beiden Screenshots verdeutlicht, dass sämtliche Attribute übereinstimmen. Dies beweist, dass ein fehlerfreier Datenaustausch zwischen den beiden Systemen etabliert wurde.

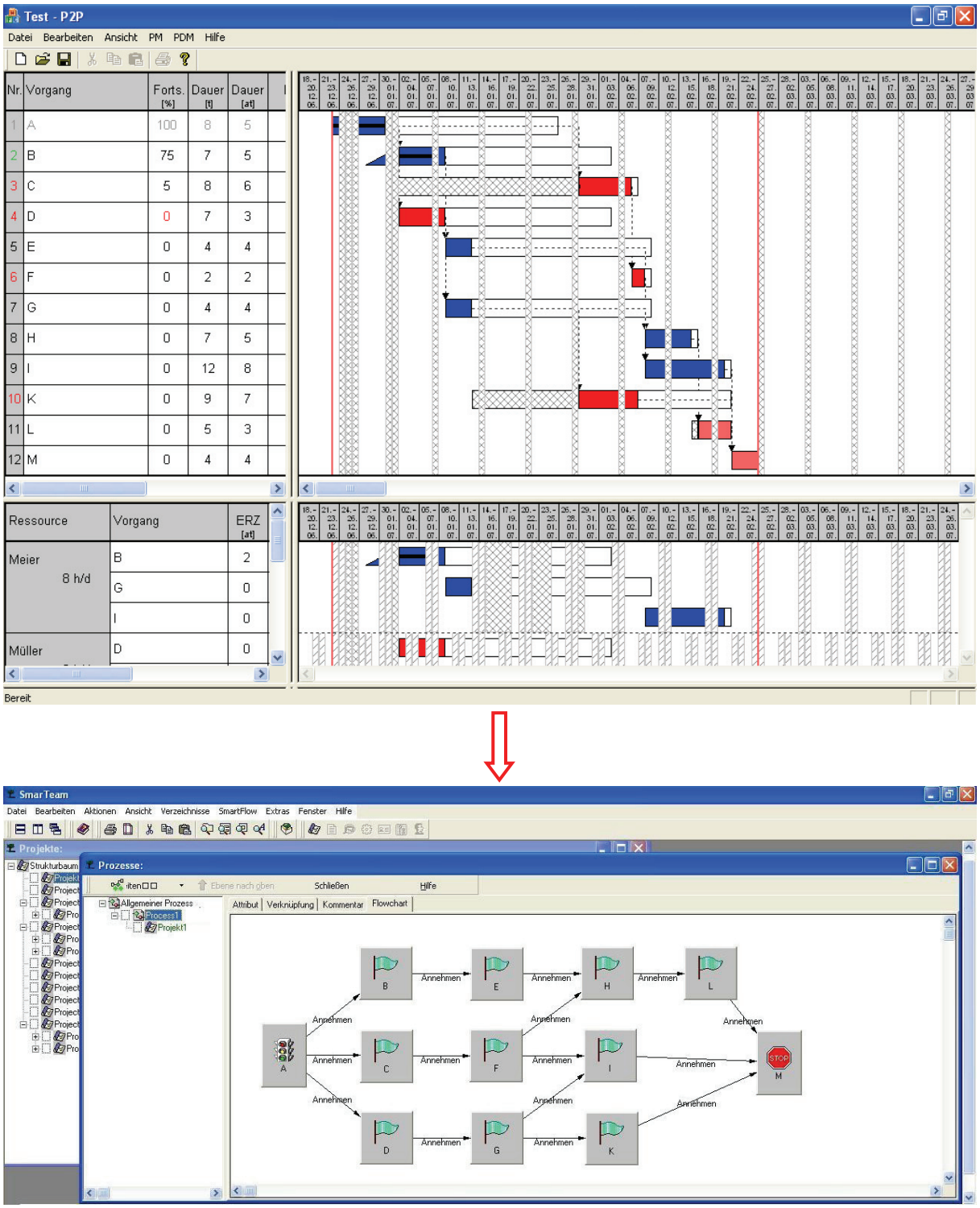


Abbildung 6-13: Beispielhafter Datenaustausch zwischen der P2P und SmarTeam

Der realisierte Prototyp unterstützt neben dem Laden und Speichern von Projekten zudem den benutzerfreundlichen Zugriff auf PDM-Dokumente. Zu diesem Zweck werden per Austauschdatei Name und IDs des in der PDM-Ansicht selektierten Dokumentes mit dem Befehl "GET_DOCS" an das PDM-System geschickt.

Danach erfolgen das Starten der dazugehörigen Applikation und das Öffnen des Dokumentes durch die Funktionen des PDM-Systems. Abbildung 6-14 zeigt ein Word-Dokument auf das von der Plantafel aus zugegriffen wurde.

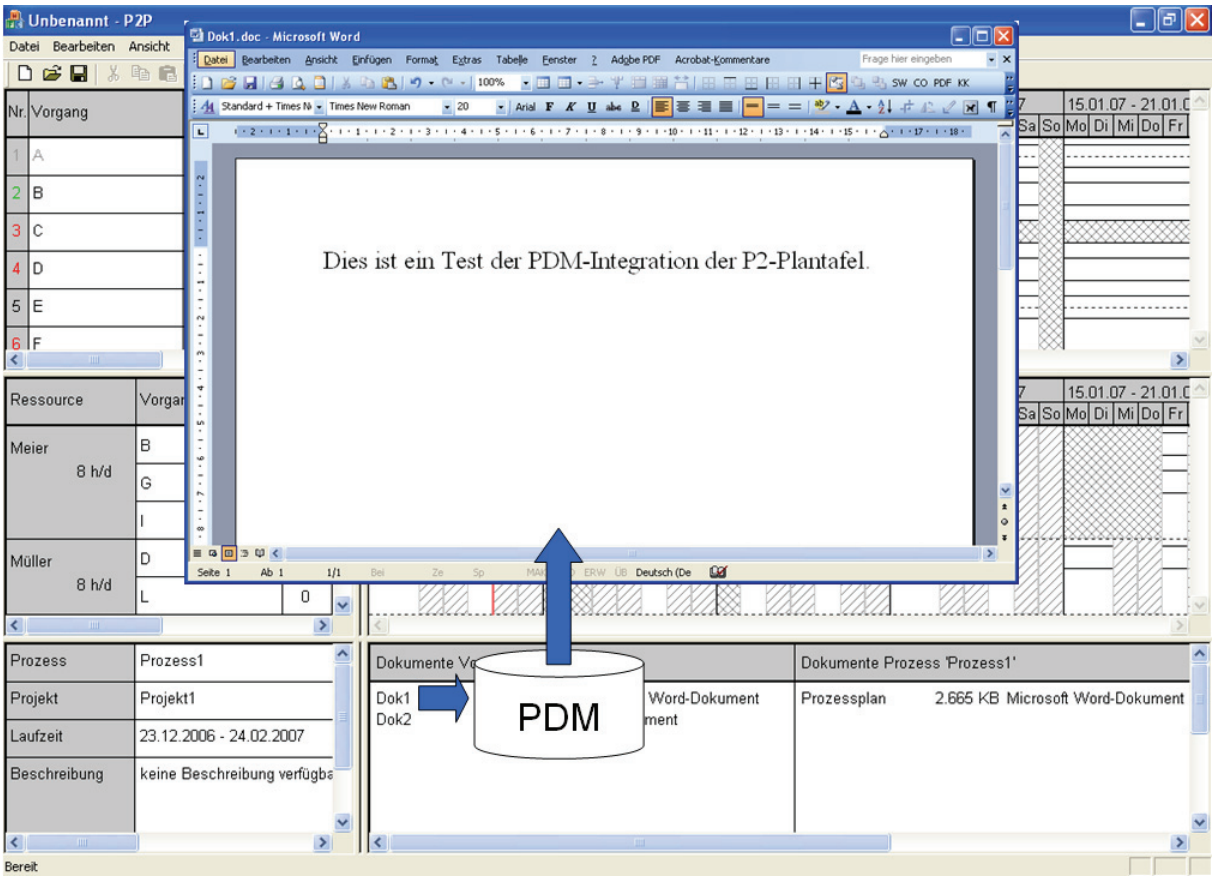


Abbildung 6-14: Word-Integration

6.2 PDM-Integration

Die Integration des Projektmanagements in das PDM-System SmarTeam wurde entsprechend der in Kapitel 5.6 dargelegten Vorgehensweise realisiert.

Durch die Integration können neben herkömmlichen PDM-Projekten nunmehr auch Projekte mit P2P-Unterstützung initialisiert werden. Projekte mit P2P-Unterstützung zeichnen sich dadurch aus, dass ihre Datenstruktur und ihre Profilkarten, zusätzlich zu den Attributen von herkömmlichen PDM-Projekten, alle für das Projektmanagement erforderlichen Attribute umfassen.
Abbildung 6-15 zeigt die Benutzeroberfläche des PDM-Systems SmarTeam. Nach der Integration kann bei dem Hinzufügen eines neuen Projektes zwischen ***Projekt (PDM)*** und ***Projekt (P2P)*** gewählt werden. Somit können je nach bei Bedarf Projekte mit oder ohne Plantafelunterstützung durchgeführt werden. Dies ist sowohl aus Gründen der Kompatibilität als auch zur Steigerung der Anwenderakzeptanz sinnvoll.

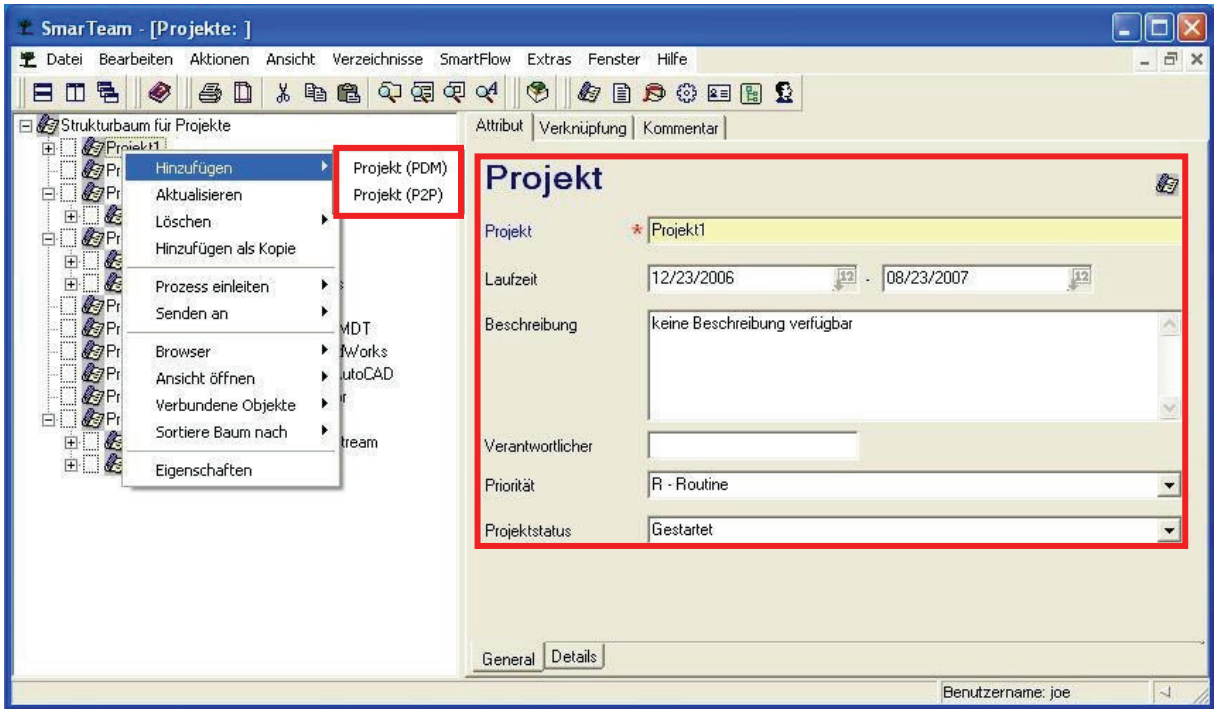


Abbildung 6-15: Kontextmenü und Profilkarten im PDM-System SmarTeam

In Abbildung 6-15 wird neben den verfügbaren Projektarten zudem eine der Profilkarten eines P2P-Projektes dargestellt. Für die Mitglieder des Projektteams sind die erweiterten Profilkarten eine wesentliche Verbesserung im Vergleich zu den bisher verfügbaren PDM-Projekten. Sie können aus den dargestellten Attributen zusätzliche Informationen, wie zum Beispiel das geplante Ende des Projektes oder einzelner Vorgänge entnehmen. Diese Informationen unterstützen das Projektteam bei der termingerechten Abwicklung des Projektes.

Die Einbettung der P2-Plantafel in die Benutzeroberfläche des PDM-Systems erfolgt über die Erweiterung der Haupt- und Kontextmenüs von SmarTeam. Abbildung 6-16 verdeutlicht, dass die Plantafel über die Menü-Funktionen ***Start P2P*** und ***Öffne Projekt in P2P*** aktiviert werden kann. Diesen SmarTeam-Funktionen wurden über die Script-Maintenance das Visual-Basic-Skript "StartP2P" zugewiesen. Wie in Kapitel 5.6.2 beschieben wurde, startet dieses Skript die P2-Plantafel und initialisiert das COM-Server-Objekt XSrv. Anschließend erfolgt eine Fallunterscheidung über welches Ereignis der Aufruf des Basic-Skripts durchgeführt wurde. Die Funktion **Start P2P** startet eine leere Plantafelanwendung. Sie kann daher jederzeit über das Hauptmenü aufgerufen werden. Währenddessen wird bei Aufruf der Funktion **Öffne Projekt in P2P** das aktive Projekt in die Plantafel importiert und geöffnet. Aus diesem Grund ist diese Funktion lediglich im Kontextmenü von P2P-Projekten verfügbar.

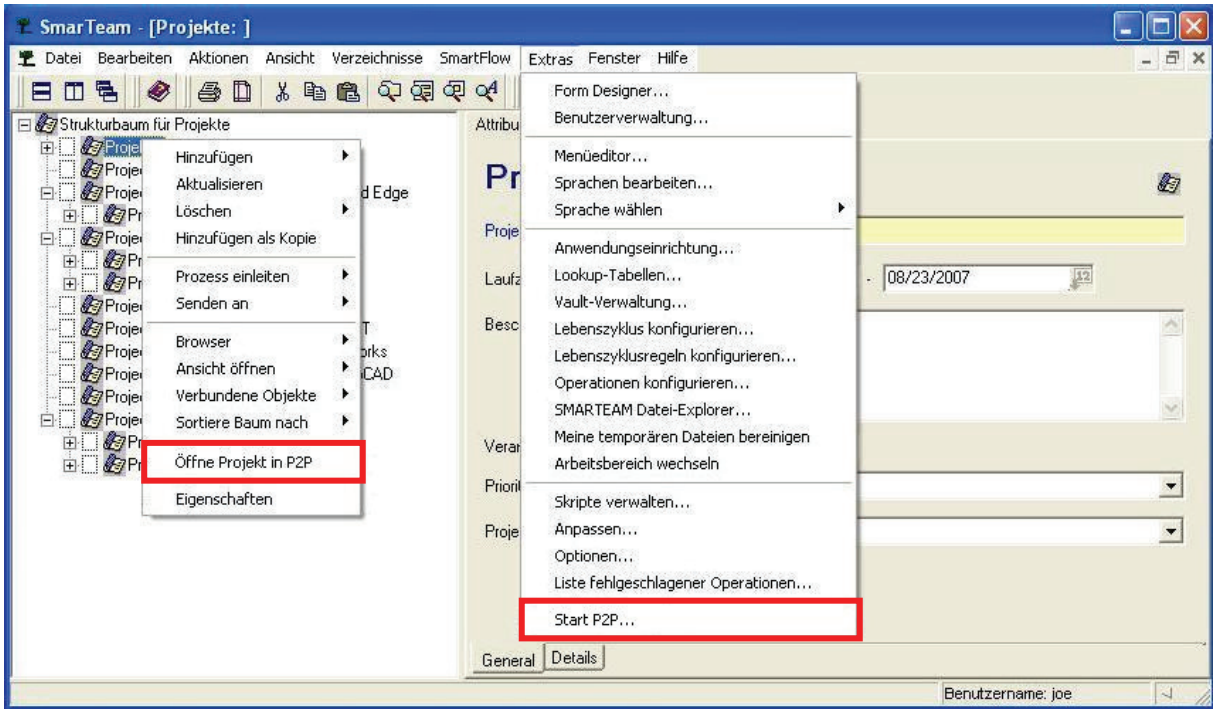


Abbildung 6-16: P2P-Integration in die PDM-GUI

Neben dem Start der P2-Plantafel aus dem PDM-System heraus, besteht zudem die Möglichkeit die Plantafel-Anwendung über das Betriebssystem zu initialisieren. In diesem Fall befindet sich die Plantafel im Stand-Alone-Modus bis eine Verbindung zu dem PDM-System etabliert wird.

7 Zusammenfassung

Die vorliegende Arbeit thematisiert die Entwicklung eines neuartigen Konzeptes zur Optimierung von Produktentwicklungsprojekten.

Vor dem Hintergrund des wachsenden globalen Wettbewerbs besteht eine zentrale Herausforderung für produzierende Unternehmen in der Minimierung der Time-To-Market, bei gleichzeitig steigenden Qualitätsanforderungen.
Die Entwicklung von innovativen Produkten unter möglichst geringem Zeitaufwand erfordert den Einsatz von EDV-Unterstützung. Eine Analyse der gegenwärtig eingesetzten EDV-Systeme zeigt, dass der Prozess der Produktentwicklung im Wesentlichen durch Produktdatenmanagement- und Projektmanagement assistiert wird. Aufgrund von unterschiedlichen Datenmodellen, redundanter Datenhaltung und des Fehlens einer zentralen Arbeitsoberfläche können derartige Systeme den Anforderungen an eine wettbewerbsfähige Produktentwicklung lediglich partiell gerecht werden. Obwohl zu beiden Systemtypen zahlreiche Anwendungen existieren, fehlt bislang ein ganzheitliches Gesamtkonzept, welches die beiden Systemtypen zu einer durchgängigen EDV-Lösung vereinigt.

Im Rahmen dieser Arbeit werden daher zunächst existierende Schwachstellen von PDM- und PM-Systemen als Einzelanwendungen aufgezeigt, bevor die Defizite der Gesamtsituation bei der EDV-Unterstützung von Produktentwicklungsprojekten ausgewertet werden.
Danach erfolgt eine Analyse der Anforderungen an ein durchgängiges Gesamtsystem, welches eine umfassende Unterstützung der gesamten Produktentwicklung ermöglicht.

Das anschließend detaillierte Konzept zeigt auf, wie diese Defizite und Anforderungen bewältigt werden können. Da existierende PDM-Systeme bereits eine funktionale Basis für einen Teil der aufgestellten Anforderungen bieten, besteht die grundlegende Architektur des konzipierten Systems aus der Erweiterung eines kommerziellen PDM-Systems und einer neu entwickelten Projektmanagement-Komponente.
Die Beschreibung des Konzeptes wird in zwei Teilschritten vollzogen. Einleitend wird die Projektmanagement-Komponente vorgestellt. Diese ist im Gegensatz zu herkömmlichen PM-Systemen auf die Anforderungen von Produktentwicklungsprojekten ausgelegt. Weiterhin werden bei der Entwicklung dieser Komponente Benutzerfreundlichkeit und Anpassungsfähigkeit als zentrale Randbedingungen berücksichtigt.
In einem zweiten Schritt erfolgt danach die Integration des Projektmanagements in ein exemplarisches PDM-System. Durch die entwickelte Systemarchitektur ist gewährleistet, dass die PM-Komponente unter geringem Anpassungsaufwand mit einem beliebigen PDM-System kombiniert werden kann.

Abschließend wird anhand einer prototypenhaften Implementierung die Realisierbarkeit des entwickelten Konzeptes nachgewiesen.

Zusammenfassend leistet die vorliegende Arbeit einen Beitrag zur Verbesserung von Planungssicherheit, Termintreue, Transparenz und Flexibilität von Produktentwicklungsprojekten.
Das entwickelte Konzept zum vollständig integrierten Produktdaten- und Projektmanagement unterstützt somit Unternehmen den gesteigerten Anforderungen an die Produktentwicklung zu begegnen und innerhalb verkürzter Durchlaufzeiten Produkte von hoher Qualität zu erstellen.

8 Literaturverzeichnis

[1] Assmann, Björn Olaf : *Herstellung hochgenauer Prototypen mittels Fräsen als quasi-generativem Rapid-Prototyping-Verfahren*. Dissertation, Universität Duisburg-Essen, Fachbereich Maschinenwesen, Institut für Prozess- und Datenmanagement, 2003

[2] Bergers, Diethard : *Produkt Engineering*. Skriptum, Universität Duisburg-Essen, Professur für Produktionstechnologie & Produktentwicklung, 2004

[3] Universität Karlsruhe (TH), Forschungszentrum Informatik: *Quantifizierbarer Nutzen Qualität.* URL http://www.pdmportal.de/index.php?id=1204 – 14.04.2008

[4] Universität Karlsruhe (TH), Forschungszentrum Informatik: *Quantifizierbarer Nutzen Zeit.* URL http://www.pdmportal.de/index.php?id=1203 – 14.04.2008

[5] Lobeck, Frank : *Konzept zur Optimierung von Produktentwicklungsprozessen einschließlich Simulation und Rapid Prototyping unter Verwendung eines neuen PLM-CAD-Integrationsmoduls*. Habilitation, Universität Duisburg-Essen, Fachbereich Maschinenwesen, Institut für Prozess- und Datenmanagement, 2003

[6] *Wikipedia : Die freie Enzyklopädie.* Ausgabe 2007/2008. Directmedia Publishing GmbH, 2007. – ISBN-13: 978-3-86640-018-4

[7] Fiedler, Rudolf: *Controlling von Projekten : Projektplanung, Projektsteuerung und Risikomanagement*. 1.Aufl. Vieweg Verlag, 2001. – ISBN 3-528-05740-8

[8] NORM VDI 2219: *Informationsverarbeitung in der Produktentwicklung : Einführung und Wirtschaftlichkeit von EDM/PDM-Systemen*

[9] Verband deutscher Maschinen- und Anlagenbau (VDMA): *Produkt Daten Management : Grundlagen und Entscheidungshilfe.* Eggebrecht-Presse KG

[10] Universität Karlsruhe (TH), Forschungszentrum Informatik: *Grundlagen.* URL http://www.plmlabor.de/index.php?id=904 – 14.04.2008

[11] Schöttner, Josef: *Produktdatenmanagement in der Fertigungsindustrie : Prinzip, Konzepte, Strategien*. 1.Aufl. Hanser Verlag, 1999. – ISBN 3-446-21152-7

[12] *Duden : Das Fremdwörterbuch*. 8. Auflage. Dudenverlag, 2005. CD-ROM

[13] Schütten, Markus: *Konzept eines COM-basierten Technischen Produktinformations-systems (TPIS)*. Dissertation, Universität Essen, Fachbereich Maschinenwesen, Institut für Prozess- und Datenmanagement, 2001

[14] Kraus, Uwe: *ERP-OnTo-PDM* : *Konzept und prototypische Realisierung einer ontologie-basierten ERP / PDM Kopplung mittels XML Technologie*. Dissertation, Universität Essen, Fachbereich Maschinenwesen, Energie- und Verfahrenstechnik, 2003

[15] Rinza, Peter : *Projektmanagement : Planung, Überwachung und Steuerung von technischen und nichttechnischen Vorhaben*. 4.Aufl. Springer Verlag, 1998. – ISBN 3-540-64021-5

[16] Seibert, Siegfried : *Technisches Management : Innovationsmanagement, Projekt-management, Qualitätsmanagement*. 1.Aufl. Teubner Verlag, 1998. – ISBN 3-519-06363-8

[17] Bergers, Diethard : *Management technischer Projekte und Geschäftsprozesse*. Skriptum, Universität Duisburg-Essen, Professur für Produktionstechnologie & Pro-duktentwicklung, 2001

[18] Haynes, Marion, E.: *Projektmanagement : Von der Idee bis zur Umsetzung*. 2.Aufl. Wirtschaftsverlag Carl Ueberreuter, 2003. – ISBN 3-8323-0999-3

[19] Corsten, Hans: *Projektmanagement*. 1.Aufl. Oldenbourg Wissenschaftsverlag, 2000. – ISBN 3-486-25252-6

[20] Meyer, Mey Mark: *Stand und Trend von Softwareunterstützung für Projektmanagement-aufgaben : Zwischenbericht zu den Ergebnissen einer Befragung von Projektmanagement-Experten*. – Universität Bremen, Institut für Projektmanagement und Innovation, 2005

[21] ASCAD GmbH, Harpener Heide 7, 44805 Bochum

[22] Unigraphics Solutions GmbH : *PLM Software : Teamcenter Project* URL http://www.ugsplm.de/produkte/teamcenter/sol_prod/project/ – 14.04.2008

[23] Wischnewski, Erik: *Modernes Projektmanagement : PC-gestützte Planung, Durchführung und Steuerung von Projekten*. 7.Aufl. Vieweg Verlag, 2001. - ISBN 3-528-65148-2

[25] Brügge, Bernd; Dutoit Allen, H.: *Objektorientierte Softwaretechnik : mit UML, Entwurfsmustern und Java.* 1.Aufl. Pearson Education, 2004. – ISBN 3-8273-7082-5

[26] WebHits: *Web-Barometer : Aktuelle Daten zur allgemeinen Nutzung von Suchmaschinen, Browsern und Betriebssystemen sowie zu den Verbreitungszahlen der wichtigsten Plug-ins. Historische Entwicklung der Marktanteile.* URL http://www.webhits.de/deutsch/index.shtml?webstats.html – 14.04.2008

[27] Heise Online: *Microsoft-Betriebssysteme dominieren weiter.* URL http://www.heise.de/newsticker/meldung/40917 – 14.04.2008

[28] Balzert, Heide: *Lehrbuch der Objektmodellierung : Analyse und Entwurf mit der UML 2.* 2.Aufl. Spektrum Verlag, 2005. – ISBN 3-8274-1162-9

[29] Budszuhn, Frank: *Visual C++ : Windows-Programmierung mit der MFC.* 1.Aufl. Addison-Wesley Verlag, 2004. – ISBN 3-8273-2175-1

[30] Scheibl, Hans-Jürgen: *Visual C++ 6.0 : für Einsteiger und Fortgeschrittene.* 1.Aufl. Hanser Verlag, 2000. – ISBN 3-446-19548-3

[31] Microsoft MSDN Library: NET oder doch lieber .NOT? URL http://msdn2.microsoft.com/de-de/library/bb978931.aspx – 14.04.2008

[32] Kruglinski, David; Sheperd, George; Wingo, Scot: *Inside Visual C++ 6.0 : Das Microsoft-Standardwerk zur Programmierung mit Visual C++: MFC, ATL, Internet und vieles mehr.* 1.Aufl. Microsoft Press, 1998. – ISBN 3-86063-461-5

[33] Hofmann, Ralf : Aufbau, Transfer und Nutzung von Wissen und dessen Anwendung im Bereich der IT-Unternehmensberatungen. Dissertation, Universität Duisburg-Essen, Fachbereich Maschinenwesen, Institut für Prozess- und Datenmanagement, 2003

[34] Lobeck, Frank: *Konzept für ein objektorientiertes, bereichsübergreifendes Dokumenteninformations- und –verwaltungssystem*. Dissertation, Universität Essen, Fachbereich Maschinenwesen, Institut für Prozess- und Datenmanagement, 1999

[35] Bergsmann, Johannes : *Nicht-funktionale Anforderungen*. Quality Knowledge Letter, Ausgabe 2004/5. Software Quality Lab. – URL: http://www.software-quality-lab.at/swql/uploads/media/SWQL-Newsletter-200412_-_Nichtfunktionale_Q-Anforderu-ngen_01.pdf – 14.04.2008

[36] Beer, Wolfgang; Birngruber, Dietrich; Mössenböck, Hanspeter; Prähofer, Herbert; Wöß, Albrecht: Die .net-Technologie : Grundlagen und Anwendungsprogrammierung. 2.Aufl. Dpunkt Verlag, 2006 – ISBN 3-89864-421-9

[37] NORMEN DIN EN ISO 9000-Reihe: *Qualitätsmanagement*

[38] NORM DIN 9126: *Bewerten von Softwareprodukten – Qualitätsmerkmale und Leitfaden zu ihrer Verwendung*

[39] NORM DIN 9241: *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten*

[40] NORM DIN 69900: *Netzplantechnik*

[41] NORM DIN 69901: *Projektmanagement*

[42] *Duden : Die deutsche Rechtschreibung*. 24. Auflage. Dudenverlag, 2006. CD-ROM

[43] Hecker, Michael: *Entwicklung eines Konzeptes für die Erstellung und Verwaltung von technischen Dokumentationen auf der Basis einer integrierten Produktentwicklung*. Dissertation, Universität Duisburg-Essen, Fachbereich Maschinenwesen, Institut für Ingenieurinformatik, 2007

[44] Herud, Jan: *Konzeption eines Software-Support-Management-Systems (SSMS) zur Verbesserung von kundenorientierter Softwarepflege und Weiterentwicklung sowie zur Wissensdistribution*. Dissertation, Universität Duisburg-Essen, Fachbereich Maschinen-wesen, Institut für Ingenieurinformatik, 2007

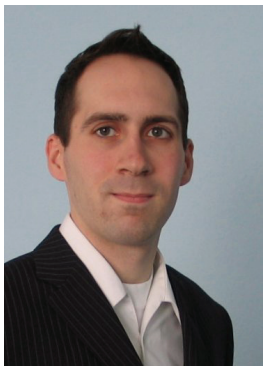
[45] Kornek, Marcus: *.NET-Framework.* TU München, 2004. URL http://www13.informatik.tu-muenchen.de/lehre/seminare/WS0304/hauptsem/Ausarbeitung06.pdf

[46] Smarteam Ltd.: *SmarTeam Object Model – Programmers Guide*, SmarTeam Corporation Ltd, 2002

[47] Stracke, Hans-Joachim: *Betriebsdatenverarbeitung I : PDM/EDM.* Skriptum, Universität Duisburg-Essen, Institut für Ingenieurinformatik, 2003

[48] Willms, Andre' : *Das C++ Codebook.* 1.Aufl. Addison-Wesley Verlag, 2003. – ISBN 3-8273-2083-6

Lebenslauf



Persönliche Daten

Name:	Dominik Raab
Geburtsdatum /-ort:	22.06.1980, Oberhausen
Familienstand:	ledig
Staatsangehörigkeit:	deutsch

Beruflicher Werdegang

seit 07/2008 — **Akademischer Rat, Universität Duisburg-Essen**
- Lehrstuhl für Mechanik und Robotik

07/2005 - 06/2008 — **Wissenschaftlicher Mitarbeiter, Universität Duisburg-Essen**
- Lehrstuhl für Mechanik und Robotik / Lehramt für berufliche Fachrichtungen (05/2007 - 06/2008)
- Institut für Ingenieurinformatik (07/2005 - 04/2007)

Studium

01/2006 - 12/2008 — **Promotion Maschinenbau, Universität Duisburg-Essen**
Titel: Entwicklung eines Konzeptes für die Optimierung von Produktentwicklungsprojekten auf Basis eines integrierten Produktdaten- und Projektmanagements

10/2000 - 06/2005 — **Studium Maschinenbau, Universität Duisburg-Essen**
Studienschwerpunkt: Produktengineering

Wehrdienst

1999 - 2000 — **Pionierbataillon in Emmerich**

Schulbildung

1990 - 1999 — **Heinrich Böll Gesamtschule in Oberhausen**

1986 - 1990 — **Grundschule Schmachtendorf in Oberhausen**

Essen, den 08.12.2008